Introduction to Visual Motion Analysis

CSC2503: Foundations of Computer Vision

David J. Fleet [1]

October 11, 2004

There are a great variety of applications that depend on analyzing the motion in image sequences. These include motion detection for surveillance, image sequence data compression (MPEG), image understanding (motion-based segmentation, recognition, depth/structure from motion), obstacle avoidance, image registration and compositing. The first step in processing image sequences is typically image velocity estimation. The result is called the optical flow field, a collection of two-dimensional velocity vectors, one for each small region (potentially, one for each pixel) of the image.

Image velocities can be measured using correlation or block-matching (e.g, see Anandan, 1989) in which each small patch of the image at one time is compared with nearby patches in the next frame. Feature extraction and matching is another way to measure the flow field (for reviews of feature tracking methods see Barron, 1984, or Aggarwal and Nandhakumar, 1988). Gradient-based algorithms are a third approach to measuring flow fields (for example, Horn and Schunk, 1981; Lucas and Kanade, 1981; Nagel, 1987). A fourth approach using spatiotemporal filtering has also been proposed (for example, Watson and Ahumada, 1985; Heeger, 1987; Grzywacz and Yuille, 1990; Fleet and Jepson, 1990). This handout concentrates on the filter-based and gradient-based methods. Emphasis is placed on the importance of multiscale, coarse-to-fine, refinement of the velocity estimates, with linear subspace (or parametric) motion models. We also discuss several other, somewhat more complex techniques, including phase-based methods, the use of robust estimation, and the use of probabilistic mixture models for computing multiple motions. For a good overview and comparison of the basic optical flow techniques, see (Barron et al, 1994).

The fields of visual motion estimation, tracking and the inference of structure from motion have extremely large bodies of literature. This introduction is mainly focused on the estimation of image motion. It does not address the related problems of visual tracking and structure from motion. Moreover, although we motivate 2D velocity estimation in terms of inferring the 3D time-varying structure of the scene, the general problem of motion estimation extends well beyond that. There are a wide variety of problems in vision and image processing for which one wants to find points in one image that *correspond* to points in the other.

For example, for stereoscopic depth perception one wants to solve this correspondence problem between the two (or more) stereo views of the scene. For conventional stereo, with cameras close together, the correspondence problem is very much like the problem of estimation the motion field. Sometimes, as in wide baseline stereo, the cameras are very far apart. In those cases the effective displacements of points from one view to the next can be very large. The correspondence problem in these cases is often referred to as a *long range motion problem.* Long range motion problems also exist in image sequences where, for example, objects are occluded for several frames and then reappear.

There are many other types of image registration problems closely related to the motion estimation problem discussed in these notes. One example concerns image mosaicing (pasting together

[1]Based on notes originally written with David Heeger at Stanford University

images of the same scene from different viewpoints). Another involves stop-frame animation for special effects. In this case many cameras view a time-varying scene at the same time. As a result they can 'stop' the scene and smoothly interpolate between views which makes it appear as though a camera is rotating around an object suspended in time (this was first used effectively in the movie "The Matrix"). Yet another source of registration problems occurs in medical imaging. Here one wants to register different imagining modalities (e.g., xray and MRI), especially in real time for computer-assisted surgery.

Now back to geometry, time-varying 3D scenes, and the properties of 2D image motion.

1 Geometry: 3D Velocity and 2D Image Velocity

Motion occurs in many applications, and there are many tasks for which motion information might be very useful. Here we concentrate on natural image sequences of 3D scenes in which objects and the camera may be moving. Typical issues for which we want computational solutions include inferring the relative 3D motion between the camera and objects in the scene, inferring the depth and surface structure of the scene, and segmenting the scene using the motion information. Towards this end, we are interested in knowing the geometric relationships between 3D motion, surface structure, and 2D image velocity.

To begin, consider a point on the surface of an object. We will represent 3D surface points as position vectors $\vec{\mathbf{X}} = (X, Y, Z)^T$ relative to a viewer-centered coordinate frame as depicted in Fig. 1. When the camera moves, or when the object moves, this point moves along a 3D path $\vec{\mathbf{X}}(t) = (X(t), Y(t), Z(t))^T$, relative to our viewer-centered coordinate frame. The instantaneous 3D velocity of the point is the derivative of this path with respect to time:

$$\vec{V} = \frac{d\vec{\mathbf{X}}(t)}{dt} = \left(\frac{dX}{dt}, \frac{dY}{dt}, \frac{dZ}{dt}\right)^T \tag{1}$$

We are interested in the image locations to which the 3D point projects as a function of time. Under perspective projection, the point $\vec{\mathbf{X}}$ projects to the image point $(x, y)^T$ given by

$$\begin{aligned} x &= fX/Z \\ y &= fY/Z, \end{aligned} \tag{2}$$

where f is the "focal length" of the projection. As the 3D point moves through time, its corresponding 2D image point traces out a 2D path $(x(t), y(t))^T$, the derivative of which is the velocity of the image of the 3D point:

$$\vec{\mathbf{u}} = \left(\frac{dx(t)}{dt}, \frac{dy(t)}{dt}\right)^T \tag{3}$$

We can combine Eqs. (2) and 3 and thereby write image velocity in terms of the 3D position and velocity of the surface point:

$$\vec{\mathbf{u}} = \frac{1}{Z}\left(\frac{dX}{dt}, \frac{dY}{dt}\right)^T - \frac{1}{Z^2}\frac{dZ}{dt}(X(t), Y(t))^T \tag{4}$$

Each visible surface point traces out a 3D path, which projects onto the image plane to form a 2D path. If one considers the paths of all visible 3D surface points, and their projections onto the

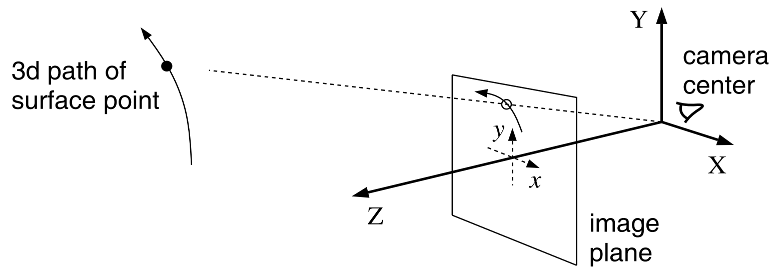


Figure 1: Camera centered coordinate frame and perspective projection. Owing to motion between the camera and the scene, a 3D surface point traverses a path in 3D. Under perspective projection, this path projects onto a 2D path in the image plane, the temporal derivative of which is called 2D velocity. The 2D velocities associated with all visible points defines a dense 2D vector field called the 2D motion field.

---

image plane, then one obtains a dense set of 2D paths, the temporal derivative of which is the vector field of 2D velocities, commonly known as the *2D motion field.*

To understand the structure of the optical flow field in greater detail it is often helpful to make further assumptions about the structure of the 3D surface or about the form of the 3D motion. We begin by considering the special case in which the objects in the scene move rigidly with respect to the camera, as though the camera was moving through a stationary scene (Longuet-Higgins and Prazdny, 1980; Bruss and Horn, 1983; Waxman and Ullman, 1985; Heeger and Jepson, 1992). This is a special case because all points on a rigid body share the same six motion parameters relative to the camera-centered coordinate frame. In particular, the instantaneous velocity of the camera through a stationary scene can be expressed in terms of the camera's 3D translation $\vec{T} = (T_x, T_y, T_z)^T$, and its instantaneous 3D rotation $\vec{\Omega} = (\Omega_x, \Omega_y, \Omega_z)^T$. Here, the direction of $\vec{\Omega}$ gives the axis of rotation while $|\vec{\Omega}|$ is the magnitude of rotation per unit time. Given this motion of the camera, the instantaneous 3D velocity of a surface point in camera-centered coordinates is

$$\left(\frac{dX}{dt}, \frac{dY}{dt}, \frac{dZ}{dt}\right)^T = -\left(\vec{\Omega} \times \vec{\mathbf{X}} + \vec{T}\right) \ . \tag{5}$$

The 3D velocities of all surface points in a stationary scene depend on the same rigid-body motion parameters, and are given by Eq. (5). When we substitute these 3D velocities for $d\vec{\mathbf{X}}/dt$ in Eq. (4) we obtain an expression for the form of the 2D motion field for a rigid scene:

$$\vec{\mathbf{u}}(x,y) = p(x,y)\mathbf{A}(x,y)\vec{T} + \mathbf{B}(x,y)\vec{\Omega} \tag{6}$$

where $p(x,y) = 1/Z(x,y)$ is inverse depth at each image location, and

$$\begin{aligned} \mathbf{A}(x,y) &= \begin{bmatrix} -f & 0 & x \\ 0 & -f & y \end{bmatrix} \\ \mathbf{B}(x,y) &= \begin{bmatrix} (xy)/f & -(f + x^2/f) & y \\ f + y^2/f & -(xy)/f & -x \end{bmatrix}. \end{aligned}$$

The matrices $\mathbf{A}(x,y)$ and $\mathbf{B}(x,y)$ depend only on the image position and the focal length.

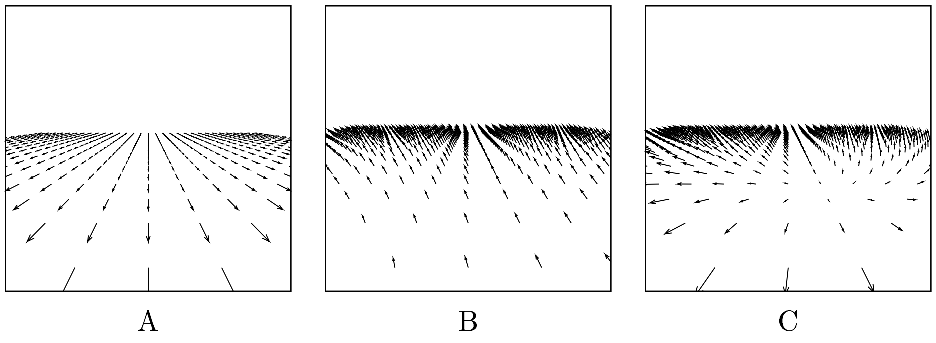


Figure 2: Example flow fields. **A:** Camera translation. **B:** Camera rotation. **C:** Translation plus rotation. Each flow vector in C is the vector sum of the two corresponding vectors in A and B.

Equation (6) describes the flow field as a function of 3D motion and depth. It has two terms. The first term is referred to as the translational component of the flow field since it depends on 3D translation and 3D depth. The second term is referred to as the rotational component since it depends only on 3D rotation. Since $p(x,y)$ (the inverse depth) and $\vec{T}$ (the translation) are multiplied together in Eq. (6), a larger distance (smaller p) or a slower 3D translation (smaller $|\vec{T}|$) both yield a slower image velocity.

Figure 2 shows some example flow fields. Each vector represents the speed and direction of motion for each local image region. Figure 2A is a flow field resulting from camera translation above a planar surface. Figure 2B is a flow field resulting from camera rotation, and Fig. 2C is a flow field resulting from simultaneous translation and rotation. Each flow vector in Fig. 2C is the vector sum of the two corresponding flow vectors is Figs. 2A and B.

Pure Translation. When a camera is undergoing a pure translational motion, features in the image move toward or away from a single point in the image, called the focus of expansion (FOE). The FOE in Fig. 2A is centered just above the horizon.

When the rotation is zero ($\vec{\Omega} = \vec{0}$),

$$\vec{\mathbf{u}}(x,y) = p(x,y)\mathbf{A}(x,y)\vec{T}$$

i.e.,

$$\begin{aligned} \mathbf{u}_1(x,y) &= p(x,y)\,(x - x_0)T_z \qquad & x_0 &= fT_x/T_z \\ \mathbf{u}_2(x,y) &= p(x,y)\,(y - y_0)T_z \qquad & y_0 &= fT_y/T_z. \end{aligned} \tag{7}$$

The image velocity is zero at image position $(x_0, y_0)^T$; this is the focus of expansion. At any other image position, the flow vector is proportional to $(x,y)^T - (x_0,y_0)^T$, that is, the flow vector points either toward or away from the focus of expansion.

Pure Rotation. When the camera is undergoing a pure rotational motion, the resulting flow field depends only on the axis and speed of rotation, independent of the scene depth/structure (see Fig. 2B for an example). Specifically, rotation results in a quadratic flow field, that is, each component $\mathbf{u}_1$ and $\mathbf{u}_2$ of the flow field is a quadratic function of image position. This is evident from the form of $\mathbf{B}(x,y)$ which contains quadratic terms in x and y (e.g., x^2, xy, etc.).

Motion With Respect to a Planar Surface. When the camera is rotating as well as translating, the flow field can be quite complex. Unlike the pure translation case, the singularity in the flow field no longer corresponds to the translation direction. But for planar surfaces, the flow field simplifies to again be a quadratic function of image position (see Fig. 2C for an example).

A planar surface can be expressed in the camera-centered coordinate frame as:

$$Z = m_x X + m_y Y + Z_0.$$

The depth map is obtained by substituting from Eq. (2) to give:

$$Z(x,y) = m_x(x/f)Z(x,y) + m_y(y/f)Z(x,y) + Z_0.$$

Solving for $Z(x,y)$:

$$Z(x,y) = \frac{f\,Z_0}{f - xm_x - ym_y},$$

and the inverse depth map is:

$$p(x,y) = \left(\frac{f}{fZ_0}\right) - \left(\frac{m_x}{fZ_0}\right)x - \left(\frac{m_y}{fZ_0}\right)y \tag{8}$$

That is, for a planar surface, the inverse depth map is also linear in x and y.

Combining Eqs. (6) and (8) gives:

$$\vec{\mathbf{u}}(x,y) = \left(\frac{1 - m_x x - m_y y}{fZ_0}\right) \begin{bmatrix} -f & 0 & x \\ 0 & -f & y \end{bmatrix} \vec{T} + \mathbf{B}\,\vec{\Omega}. \tag{9}$$

We already know (see above) that the rotational component of the flow field (the second term, $\mathbf{B}\vec{\Omega}$) is quadratic in x and y. It is clear that the translational component (the first term) in the above equation is also quadratic in x and y.

For curved 3D surfaces the translational component, and hence the 2D motion field, will be cubic or even higher order. The rotational component is always quadratic.

Occlusions. The motion field is often more complex than is implied by the above equations. For example, the boundaries of objects gives rise to occlusions in an image sequence, which cause discontinuities in the optical flow field. A detailed discussion of these issues is beyond the scope of this presentation.

2 Gradient-Based 2D Motion Estimation

The 2D motion field described above is a geometric entity. We now turn to the problem of estimating it from the spatiotemporal variations in image intensity. We refer to such an estimate as a 2D *optical flow field.* Towards this end, a number of people (Horn and Schunk, 1981; Lucas and Kanade, 1981; Nagel, 1987; and others) have proposed algorithms that compute optical flow from spatial and temporal derivatives of image intensity. This is where we begin.

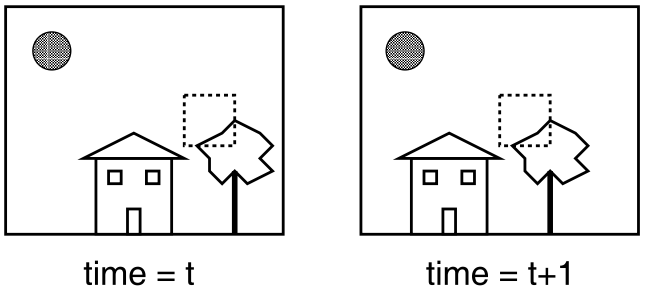


Figure 3: Intensity conservation assumption: the image intensities are shifted (locally translated) from one frame to the next.

Intensity Conservation and Block Matching. The usual starting point for velocity estimation is to assume that the intensities are shifted (locally translated) from one frame to the next, and that the shifted intensity values are *conserved*, i.e.,

$$f(x, y, t) = f(x + \mathbf{u}_1, y + \mathbf{u}_2, t + 1), \tag{10}$$

where $f(x, y, t)$ is image intensity as a function of space and time, and $\vec{\mathbf{u}} = (\mathbf{u}_1, \mathbf{u}_2)$ is velocity. Of course, this *intensity conservation* assumption is only approximately true in practice. The effective assumption is that the surface radiance remains fixed from one frame to the next in an image sequence. One can fabricate scene constraints for which this holds, but they are somewhat artificial: For example, the scene might be constrained to contain only Lambertian surfaces (no specularities), with a distant point source (so that changing the distance to the light source has no effect), with no secondary illumination effects (shadows or inter-surface reflection). Although unrealistic, it is remarkable that this intensity conservation constraint works as well as it does!

Then a typical block matching algorithm would proceed, as illustrated in Fig. 3, by choosing a region (e.g., 8x8 pixels) from one frame and shifting it over the next frame to find the best match. One might search for a peak in the cross-correlation

$$\sum f(x, y, t)\, f(x + \mathbf{u}_1, y + \mathbf{u}_2, t + 1),$$

or (equivalently) a minimum in the sum-of-squared-differences (SSD)

$$\sum [f(x, y, t) - f(x + \mathbf{u}_1, y + \mathbf{u}_2, t + 1)]^2.$$

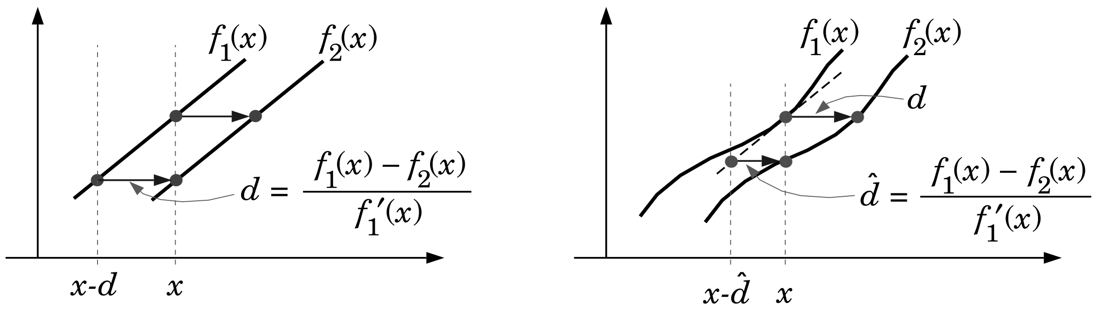
In both cases, the summation is taken over equal sized blocks in the two frames.

Gradient Constraints. To derive a simple, linear method for estimating the velocity $\mathbf{u}$, let's first digress to consider a 1-d case. Let our signals at two time instants be $f_1(x)$ and $f_2(x)$, and suppose that $f_2(x)$ is a translated version of $f_1(x)$; i.e., $f_2(x) = f_1(x - d)$ for displacement d. This situation is illustrated in Fig. 4. A Taylor series expansion of $f_1(x - d)$ about x is given by

$$f_1(x - d) \;=\; f_1(x) - d\, f_1'(x) + O(d^2 f_1'') \tag{11}$$

where f' denotes the derivative of f with respect to x. Given this expansion, we can now write the difference between the two signals at location x as

$$f_1(x) - f_2(x) \;=\; d\, f_1'(x) + O(d^2 f_1'')$$



$f_1(x)$
$f_2(x)$
$d = \frac{f_1(x) - f_2(x)}{f_1'(x)}$
$x\text{-}d$
x
$f_1(x)$
$f_2(x)$
d
$\hat{d} = \frac{f_1(x) - f_2(x)}{f_1'(x)}$
$x\text{-}\hat{d}$
x

Figure 4: The gradient constraint relates the displacement at one point of the signal to the temporal difference and the spatial derivative (slope) of the signal. For a displacement of a linear signal (left), the difference in signal values at a point divided by the slope gives the displacement. In the case of nonlinear signals (right), the difference divided by the slope gives an approximation to the displacement.

By neglecting higher-order terms, we arrive at an approximation to the displacement d, that is,

$$\hat{d} = \frac{f_1(x) - f_2(x)}{f_1'(x)} \ . \tag{12}$$

Motion Gradient Constraint. This approach generalizes straightforwardly to two spatial dimensions, to produce the *motion gradient constraint equation.* As above, one assumes that the time-varying image intensity is well approximated by a first-order Taylor series expansion,

$$f(x + \mathbf{u}_1, y + \mathbf{u}_2, t + 1) \approx f(x, y, t) + \mathbf{u}_1 f_x(x, y, t) + \mathbf{u}_2 f_y(x, y, t) + f_t(x, y, t),$$

where f_x, f_y, and f_t are the spatial and temporal partial derivatives of image intensity. If we ignore second- and higher-order terms in the Taylor series, and then substitute the resulting approximation into Eq. (10) we obtain

$$\mathbf{u}_1 f_x(x, y, t) + \mathbf{u}_2 f_y(x, y, t) + f_t(x, y, t) = 0 \ . \tag{13}$$

This equation relates the velocity at one location in the image to the spatial and temporal derivatives of image intensity at that same location. Eq. (13) is called the motion gradient constraint.

If the time between frames is relatively large, so that the temporal derivative $f_t(x, y, t)$ is unavailable, then a similar constraint can be derived by taking only the first-order spatial terms of the Taylor expansion. In this case we obtain

$$\mathbf{u}_1 f_x(x, y, t) + \mathbf{u}_2 f_y(x, y, t) + \Delta f(x, y, t) = 0 \ , \tag{14}$$

where $\Delta f(x, y, t) = f(x, y, t + 1) - f(x, y, t)$.

Intensity Conservation. A different way of deriving the gradient constraint can be found if we consider the 2D paths in the image plane, $(x(t), y(t))^T$, along which the intensity is constant. In

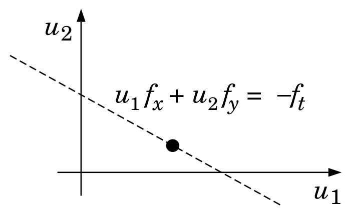


Figure 5: The gradient constraint equation constrains velocity to lie somewhere on a line in 2D velocity space. That is, $\mathbf{u}_1 f_x + \mathbf{u}_2 f_y = -f_t$ defines a line perpendicular (f_x, f_y). If one had a set of measurements from nearby points, all with different gradient directions but consistent with the same velocity, then we expect the constraint lines to intersect at the true velocity (the black dot).

---

other words, imagine that we are tracking points of constant intensity from frame to frame. Image velocity is simply the temporal derivative of such paths.

More formally, a path $(x(t), y(t))^T$ along which intensity is equal to a constant c must satisfy the equation

$$f(x(t), y(t), t) = c \ . \tag{15}$$

As explained above, assuming that the path is smooth, the temporal derivative of the path $(x(t), y(t))^T$ gives us the optical flow. To obtain a constraint on optical flow we therefore differentiate equation 15:

$$\frac{d}{dt} f(x(t), y(t), t) = 0 \tag{16}$$

The right hand side is zero because c is a constant. Because the left-hand-side is a function of 3 variables which all depend on time t, we take the total derivative and use the chain rule to obtain:

$$\begin{aligned} \frac{d}{dt} f(x(t), y(t), t) &= \frac{\partial f}{\partial x}\frac{dx}{dt} + \frac{\partial f}{\partial y}\frac{dy}{dt} + \frac{\partial f}{\partial t}\frac{dt}{dt} \\ &= f_x \mathbf{u}_1 + f_y \mathbf{u}_2 + f_t \end{aligned} \tag{17}$$

In this last step the velocities were substituted for the path derivatives (i.e., $\mathbf{u}_1 = dx/dt$ and $\mathbf{u}_2 = dy/dt$). Finally, if we combine Eqs. (16) and (17) we obtain the motion gradient constraint.

Equation (16) is often referred to as an *intensity conservation* constraint. In terms of the 3D scene, this conservation assumption implies that the image intensity associated with the projection of a 3D point does not change as the object or the camera move. This assumption is often violated to some degree in real image sequences.

Combining Constraints. It is impossible to recover velocity, given just the gradient constraint at a single position, since Eq. (13) offers only one linear constraint to solve for the two unknown components of velocity. Only the component of velocity in the gradient direction is determined. In effect, as shown in Fig 5, the gradient constraint equation constrains the velocity to lie somewhere along a line in velocity space. Further assumptions or measurements are required to constrain velocity to a point in velocity space.

Many gradient-based methods solve for velocity by combining information over a spatial region. The different gradient-based methods use different combination rules. A particularly simple rule for combining constraints from two nearby spatial positions is:

$$\begin{bmatrix} f_x(x_1,y_1,t) & f_y(x_1,y_1,t) \\ f_x(x_2,y_2,t) & f_y(x_2,y_2,t) \end{bmatrix} \begin{pmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \end{pmatrix} + \begin{bmatrix} f_t(x_1,y_1,t) \\ f_t(x_2,y_2,t) \end{bmatrix} = \vec{0}, \tag{18}$$

where the two coordinate pairs (x_i, y_i) correspond to the two spatial positions. Each row of Eq. (18) is the gradient constraint for one spatial position. Solving this equation simultaneously for both spatial positions gives the velocity that is consistent with both constraints.

A better option is to combine constraints from more than just two spatial positions. When we do this we will not be able to exactly satisfy all of the gradient constraints simultaneously because there will be more constraints than unknowns. To measure the extent to which the gradient constraints are not satisfied we square each constraint and sum them:

$$E(\mathbf{u}_1,\mathbf{u}_2) = \sum_{x,y} g(x,y)\,[\mathbf{u}_1 f_x(x,y,t) + \mathbf{u}_2 f_y(x,y,t) + f_t(x,y,t)]^2 \ , \tag{19}$$

where $g(x,y)$ is a window function that is zero outside the neighborhood within which we wish to use the constraints. Each squared term in the summation is a constraint on the flow from a different (nearby) position. Usually, we want to weight the terms in the center of the neighborhood higher to give them more influence. Accordingly, it is common to let $g(x,y)$ be a decaying window function such as a Gaussian.

Since there are now more constraints than unknowns, there may not be a solution that satisfies all of the constraints simultaneously. In other words, $E(\mathbf{u}_1,\mathbf{u}_2)$ will typically be non-zero for all $(\mathbf{u}_1,\mathbf{u}_2)$. The choice of $(\mathbf{u}_1,\mathbf{u}_2)$ that minimizes $E(\mathbf{u}_1,\mathbf{u}_2)$ is the least squares estimate of velocity.

Least-Squares Estimation. Since Eq. (19) is a quadratic expression, there is a simple analytical expression for the velocity estimate. The solution is derived by taking derivatives of Eq. (19) with respect to $\mathbf{u}_1$ and $\mathbf{u}_2$, and setting them equal to zero:

$$\begin{aligned} \frac{\partial E(\mathbf{u}_1,\mathbf{u}_2)}{\partial \mathbf{u}_1} &= \sum_{xy} g(x,y)\ \left[\mathbf{u}_1 {f_x}^2 + \mathbf{u}_2 f_x f_y + f_x f_t\right] &= 0 \\ \frac{\partial E(\mathbf{u}_1,\mathbf{u}_2)}{\partial \mathbf{u}_2} &= \sum_{xy} g(x,y)\ \left[\mathbf{u}_2 {f_y}^2 + \mathbf{u}_1 f_x f_y + f_y f_t\right] &= 0 \end{aligned}$$

These equations may be rewritten as a single equation in matrix notation:

$$\mathbf{M} \cdot \vec{\mathbf{u}} + \vec{b} = \vec{0}, \tag{20}$$

where the elements of $\mathbf{M}$ and $\vec{b}$ are:

$$\mathbf{M} = \begin{bmatrix} \sum g\, {f_x}^2 & \sum g\, f_x f_y \\ \sum g\, f_x f_y & \sum g\, {f_y}^2 \end{bmatrix}$$

$$\vec{b} = \begin{pmatrix} \sum g\, f_x f_t \\ \sum g\, f_y f_t \end{pmatrix}$$

The least-squares solution is then given by

$$\hat{\mathbf{u}} = -\mathbf{M}^{-1}\vec{b}, \tag{21}$$

presuming that $\mathbf{M}$ is invertible.

Implementation with Image Operations. Remember that we want to compute optical flow at every spatial location. Therefore we will need the matrix $\mathbf{M}$ and the vector $\vec{b}$ at each location. In particular, we should rewrite our matrix equation as

$$\mathbf{M}(x,y) \cdot \vec{\mathbf{u}}(x,y) + \vec{b}(x,y) = \vec{0} \; .$$

As above in Eq. (21), the elements in $\mathbf{M}(x,y)$ and $\vec{b}(x,y)$ at position $(x,y)^T$ are local weighted summations of the intensity derivatives. What we intend to show here is that the elements of $\mathbf{M}(x,y)$ and $\vec{b}(x,y)$ can be computed in a straightforward way using convolutions and image point-operations.

To begin, let $g(x,y)$ be a Gaussian window sampled on a square grid of width w. Then at a specific location $(x,y)^T$ the elements of the matrix $\mathbf{M}(x,y)$ and the vector $\vec{b}(x,y)$ are given by:

$$\mathbf{M} = \left[\begin{array}{cc} \sum_\alpha \sum_\beta g(\alpha,\beta) f_x^2(x-\alpha, y-\beta) & \sum_\alpha \sum_\beta g(\alpha,\beta) f_x(x-\alpha, y-\beta) f_y(x-\alpha, y-\beta) \\ \sum_\alpha \sum_\beta g(\alpha,\beta) f_x(x-\alpha, y-\beta) f_y(x-\alpha, y-\beta) & \sum_\alpha \sum_\beta g(\alpha,\beta) f_y^2(x-\alpha, y-\beta) \end{array} \right]$$

$$\vec{b} = \left(\begin{array}{c} \sum_\alpha \sum_\beta g(\alpha,\beta)\, f_x(x-\alpha, y-\beta)\, f_t(x-\alpha, y-\beta) \\ \sum_\alpha \sum_\beta g(\alpha,\beta)\, f_y(x-\alpha, y-\beta)\, f_t(x-\alpha, y-\beta) \end{array} \right)$$

where each summation is over the width of the Gaussian, from $-w$ to w. Although this notation is somewhat cumbersome, it serves to show explicitly that each element of the matrix (as a function of spatial position) is actually equal to the convolution of $g(x,y)$ with products of the intensity derivatives. To compute the elements of $\mathbf{M}(x,y)$ and $\vec{b}(x,y)$ one simply applies derivative filters to the image, takes products of their outputs, and then blurs the results with a Gaussian. For example, the upper-left component of $\mathbf{M}(x,y)$, ie. $m_{11}(x,y)$, is computed by: (1) applying an x-derivative filter, (2) squaring the result at each pixel, (3) blurring with a Gaussian lowpass filter.

Gaussian PreSmoothing. Another practicality worth mentioning here is that some preprocessing before applying the gradient constraint is generally necessary to produce reliable estimates of optical flow. The reason is primarily due to the use of a first-order Taylor series expansion in deriving the motion constraint equation. By smoothing the signal, it is hoped that we are reducing the relative amplitudes of higher-order terms in the image. Therefore, it is common to smooth the image sequence with a lowpass filter before estimating derivatives, as in Eq. (3). Often, this presmoothing can be incorporated into the derivative filters used to computer the spatial and temporal derivatives of the image intensity. This also helps avoid problems caused by small amounts of aliasing, something we'll say more about later.

Aperture Problem. When the matrix $\mathbf{M}$ in Eq. (21) is singular (or ill-conditioned), there are not enough constraints to solve for both unknowns. This always happens when the region (or aperture) over which we collect constraints for the least-squares problem gets sufficiently small, and is therefore called the *aperture problem.* In such cases there is not enough information to disambiguate the true direction of motion. In fact, for some image patterns (e.g., an extended grating or edge) the image constraints may be insufficient regardless of the aperture size.

This problem is illustrated in Fig. 6(A). All the gradient constraints within the circular aperture constrain the velocity to a 1D family of velocities that lie along the line shown in velocity space.

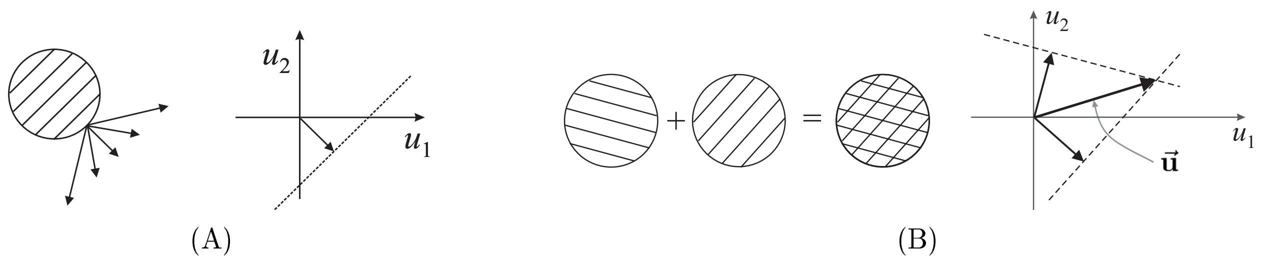


Figure 6: *Aperture problem.* (A) Single moving grating viewed through a circular aperture. The diagonal line indicates the locus of velocities compatible with the motion of the grating. (B) Plaid composed of two moving gratings. The lines give the possible motion of each grating alone. Their intersection is the only shared motion.

The best we can do is to extract only component of the unknow velocity. In particular, it only constrains the normal component of the velocity, i.e.,

$$v_n \;=\; \frac{-f_t}{||\vec{\nabla} f||}\,\frac{\vec{\nabla} f}{||\vec{\nabla} f||}\;,$$

where $\vec{\nabla} f \equiv (f_x,\, f_y)^T$. The local gradients in the region do not constraint the tangential velocity (i.e., the 2D velocity in the direction of grating orientation).

Figure 6(B) shows how the motion is disambiguated when there is more spatial structure. The plaid pattern illustrated in Fig. 6(B) is composed of two moving gratings. The lines give the possible motion of each grating alone. Their intersection is the only 2D velocity consistent with both constraint lines. More precisely, the combination of gradient constraints according to the summation in Eq. (18) is related to the intersection of constraints rule depicted in Fig. 6(B). The gradient constraint, Eq. (13), is linear in both $\mathbf{u}_1$ and $\mathbf{u}_2$. Given measurements of the derivatives, (f_x, f_y, f_t), there is a line of possible solutions for $(\mathbf{u}_1, \mathbf{u}_2)$, analogous to the constraint line illustrated in Fig. 6(left). For each different position, there will generally be a different constraint line. Equation (18) gives the intersection of these constraint lines, analogous to Fig. 6(B).

Of course, the important issue here is not due soley to the aperture size. Rather, it is really about the dimensionality of the image structure. The image may be 1-dimensional over large regions in which case even for large apertures one cannot uniquely determine the 2D velocity. But the problem is even more complex; i.e., while one might want a sufficiently large aperture to include constraints with different orientations, one must also keep the aperture sufficiently small that the assumption that there is a single smooth motion in the aperture also holds. This has been referred to as the *generalized aperture problem.*

In practice, to ensure the reliability of the velocity estimates it is common to check the condition number of $\mathbf{M}$ (or better yet, make sure that the magnitude of the smallest singular value of $\mathbf{M}$ is large enough). In order to ensure that the constraint appear consistent with a single smooth motion, one can check the residual error of the least-squares estimate.

Regularization Solution. Another way to add further constraints to the gradient constraint equation is to invoke a smoothness assumption. In other words, one can constrain (regularize) the estimation problem by assuming that resultant flow field should be the smoothest flow field that is also consistent with the gradient constraints. The central assumption here, like that made with the area-based regression approach above, is that the flow in most image regions is expected to be smooth because smooth surfaces generally give rise to smooth optical flow fields.

For example, let's assume that we want the optical flow to minimize any violations of the gradient-constraint equation, and at the same time, to minimize the magnitude of velocity changes between neighboring pixels. For example, we may wish to minimize:

$$E \;=\; \sum_{x,y}(f_x\mathbf{u}_1 + f_y\mathbf{u}_2 + f_t)^2 \;+\; \lambda \sum_{x,y}\left(\left(\frac{\partial \mathbf{u}_1}{\partial x}\right)^2 + \left(\frac{\partial \mathbf{u}_1}{\partial y}\right)^2 + \left(\frac{\partial \mathbf{u}_2}{\partial x}\right)^2 + \left(\frac{\partial \mathbf{u}_2}{\partial y}\right)^2\right)$$

In practice, the derivatives are approximated by numerical derivative operators, and the resulting minimization problem has as its solution a large system of linear equations. Because of the large dimension of the linear system, it is commonly solved iteratively using *Jacobi*, *Gauss-Seidel*, or *conjugate-gradient* iteration.

One of the main disadvantages of this approach is the extensive amount of smoothing often imposed, even when the actual optical flow field is not smooth. Some researchers have advocated the use of *line processes* that allow for violations of smoothness at certain locations where the flow field constraints indicate discontinuous flow (due to occlusion for example). Such recent regularized solutions achieve the same accuracy as area-based regression techniques. This issue is somewhat involved and beyond the scope of the handout.

3 Space-Time Filters and the Frequency Domain

One can also view image motion and these approaches to estimating optical flow in the frequency domain. This will help us to understand aliasing, and it also serves as the foundation for filter-based methods for optical flow estimation. In addition, a number of models of biological motion sensing have been based on direction selective, spatiotemporal linear operators (e.g., Watson and Ahumada, 1985; Adelson and Bergen, 1985; Heeger, 1987; Grzywacz and Yuille, 1990). The concept underlying these models is that visual motion is like orientation in space-time, and that spatiotemporally-oriented, linear operators can be used to detect and measure it.

Figure 7 shows a simple example. Figure 7A depicts a vertical bar moving to the right. Imagine that we stack the consecutive frames of this image sequence one after the next. We end up with a three-dimensional volume (space-time cube) of intensity data like that shown in Figure 7B. Figure 7C shows an x-t slice through this space-time cube. The slope of the edges in the x-t slice equals the horizontal component of the bar's velocity (change in position over time). Different speeds correspond to different slopes.

Following Adelson and Bergen (1986), and Simoncelli (1993) we now show that the gradient-based solution can be expressed in terms of the outputs of a set of space-time oriented linear filters. To this end, note that the derivative operations in the gradient constraint may be written as convolutions. Furthermore, we can prefilter the stimuli to extract some spatiotemporal subband, and perform the analysis on that subband. Consider, for example, prefiltering with a space-time Gaussian function. We can define the derivative of the smoothed image as

$$\hat{f}_x(x,y,t) \;\equiv\; \frac{\partial}{\partial x}[g(x,y,t) * f(x,y,t)] \;=\; g_x(x,y,t) * f(x,y,t),$$

where $*$ is convolution and g_x is the x-derivative of a Gaussian. In words, we compute $\hat{f}_x$ by convolving with $g_x = \partial g/\partial x$, a spatiotemporal linear filter. We compute $\hat{f}_y$ and $\hat{f}_t$ similarly.

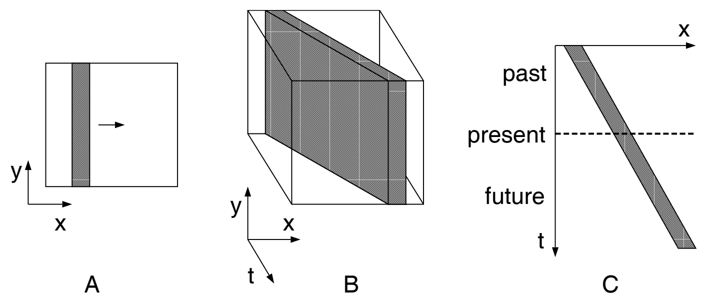


Figure 7: Orientation in space-time (based on an illustration by Adelson and Bergen, 1985). **A:** A vertical bar translating to the right. **B:** The space-time cube of image intensities corresponding to motion of the vertical bar. **C:** An x-t slice through the space-time cube. Orientation in the x-t slice is the horizontal component of velocity. Motion is like orientation in space-time, and spatiotemporally oriented filters can be used to detect and measure it.

Then the gradient constraint can be rewritten as:

$$(\mathbf{u}_1 g_x + \mathbf{u}_2 g_y + g_t) * f = 0, \tag{22}$$

where g_x, g_y, and g_t are the space-time derivative filters. For a fixed $\vec{\mathbf{u}}$, Eq. (22) is the convolution of a space-time filter, namely $(\mathbf{u}_1 g_x + \mathbf{u}_2 g_y + g_t)$, with the image sequence, $f(x, y, t)$. This filter is a weighted sum of the three "basis" derivative filters (g_x, g_y, and g_t), and hence, is itself a directional derivative in some space-time direction. The gradient constraint states that there is some space-time orientation for which the directional derivative is zero. This filter-based interpretation of the gradient constraint is depicted (in one spatial dimension) in Fig. 8.

Frequency Domain Analysis of the Gradient Constraint. The gradient constraint for image velocity estimation also has a simple interpretation as regression in the spatiotemporal Frequency domain.

The power spectrum of a translating two-dimensional pattern lies on a plane in the Fourier domain (Watson and Ahumada, 1983, 1985; Fleet, 1992), and the tilt of this plane depends on velocity. An intuition for this result can be obtained by first considering each spatiotemporal frequency component separately. The spatial frequency of a moving sine grating is expressed in cycles per pixel and its temporal frequency is expressed in cycles per frame. Velocity, which is distance over time (or pixels per frame), equals the temporal frequency divided by the spatial frequency. More formally, a drifting 1d sinusoidal grating can be expressed in terms of spatial frequency and velocity as

$$f(x, t) = \sin(\omega_x (x - \mathbf{u}_1 t)) \tag{23}$$

where ω_x is the spatial frequency of the grating and $\mathbf{u}_1$ is the velocity. It can also be expressed in terms of spatial frequency ω_x and temporal frequency ω_t as

$$f(x, t) = \sin(\omega_x x + \omega_t t) \tag{24}$$

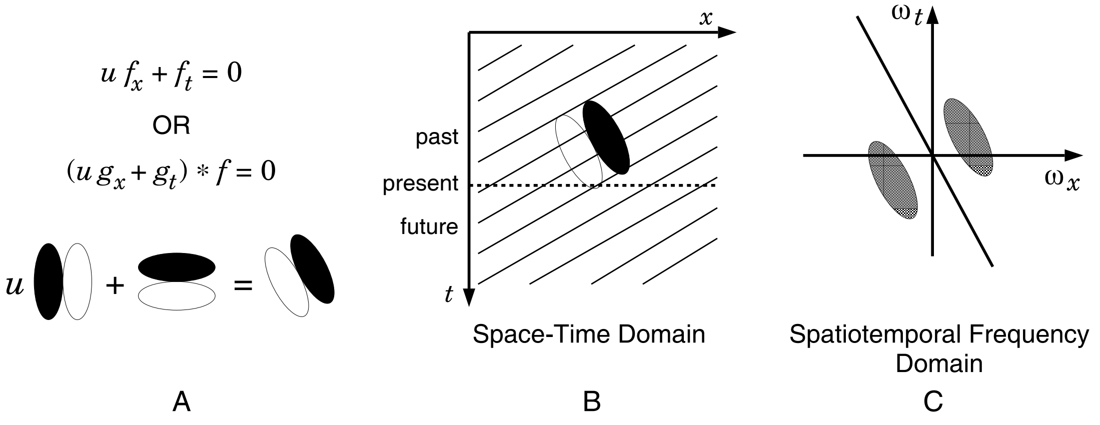


Figure 8: Space-time filters and the gradient constraint. **A:** The gradient constraint states that there is some space-time orientation for which the directional derivative is zero. **B:** Pattern moving from right to left and the space-time derivative filter with the orthogonal space-time orientation so that convolving the two gives a zero response. **C:** Spatiotemporal Fourier transform of **B**. The power spectrum of the translating pattern (the line passing through the origin) and the frequency response of the derivative filter (the pair of symmetrically positioned blobs); the product of the two is zero.

The relation between frequency and velocity can be derived algebraically by setting $\omega_x(x - \mathbf{u}_1 t) = \omega_x x + \omega_t t$. From this one can show that

$$\mathbf{u}_1 = -\omega_t/\omega_x \tag{25}$$

Now consider a one-dimensional spatial pattern that has many spatial-frequency components (e.g., made up of parallel bars, edges, or random stripes), translating with speed $\mathbf{u}_1$. Each frequency component has a spatial frequency ω_x and a temporal frequency ω_t but all of the Fourier components share the same velocity so each component satisfies

$$\mathbf{u}_1\,\omega_x + \omega_t = 0.$$

Analogously, two dimensional patterns (arbitrary textures) translating in the image plane occupy a plane in the spatiotemporal frequency domain:

$$\mathbf{u}_1\,\omega_x + \mathbf{u}_2\,\omega_y + \omega_t = 0.$$

For example, the expected value of the power spectrum of a translating white noise texture is a constant within this plane and zero outside of it. The plane is perpendicular to the vector $(\mathbf{u}_1, \mathbf{u}_2, 1)^T$.

Formally, consider a space-time image sequence $f(x, y, t)$ that we construct by translating a 2D image $f_0(x, y)$ with a constant velocity $\vec{\mathbf{u}}$. The image intensity over time may be written as

$$f(x, y, t) = f_0(x - \mathbf{u}_1 t,\, y - \mathbf{u}_2 t).$$

At time $t = 0$ the spatiotemporal image is equal to $f_0(x, y)$. The three-dimensional Fourier transform of $f(x, y, t)$ is:

$$F(\omega_x, \omega_y, \omega_t) = \int\int\int_{-\infty}^{\infty} f(x, y, t) e^{-j2\pi(x\omega_x + y\omega_y + t\omega_t)} dx dy dt. \tag{26}$$

To simplify this expression, we substitute $f_0(x - \mathbf{u}_1 t, y - \mathbf{u}_2 t)$ for $f(x, y, t)$ and arrange the order of integration so that time is integrated last:

$$F(\omega_x, \omega_y, \omega_t) = \int \left[\int\int_{-\infty}^{\infty} f_0(x - \mathbf{u}_1 t, y - \mathbf{u}_2 t) e^{-j2\pi(x\omega_x + y\omega_y)} dx dy \right] e^{-j2\pi t\omega_t} dt. \tag{27}$$

The term inside the square brackets is the 2D Fourier transform of $f_0(x - \mathbf{u}_1 t,\ y - \mathbf{u}_2 t)$ which can be simplified using the Fourier shift property to yield

$$\begin{aligned} F(\omega_x, \omega_y, \omega_t) &= \int_{-\infty}^{\infty} F_0(\omega_x, \omega_y) e^{-j2\pi(\mathbf{u}_1 t\omega_x + \mathbf{u}_2 t\omega_y)} e^{-j2\pi t\omega_t} dt \\ &= F_0(\omega_x, \omega_y) \int_{-\infty}^{\infty} e^{-j2\pi t(\mathbf{u}_1 \omega_x + \mathbf{u}_2 \omega_y)} e^{-j2\pi t\omega_t} dt \end{aligned} \tag{28}$$

where $F_0(\omega_x, \omega_y)$ is the 2D Fourier transform of $f_0(x, y)$. The Fourier transform of a complex exponential is a Dirac delta function, and therefore this expression simplifies further to

$$F(\omega_x, \omega_y, \omega_t) = F_0(\omega_x, \omega_y)\, \delta(\mathbf{u}_1 \omega_x + \mathbf{u}_2 \omega_y + \omega_t)$$

This equation shows that the Fourier transform is only nonzero on a plane, because the delta function is only nonzero when $\mathbf{u}_1 \omega_x + \mathbf{u}_2 \omega_y + \omega_t = 0$.

There are many important reasons to understand motion in the frequency domain. For example, if the motion of an image region may be approximated by translation in the image plane then the velocity of the region may be computed by finding the plane (in the Fourier domain) in which all the power resides. Figure 8 illustrates how a derivative filter can be used to determine which plane contains the power. The frequency response of a spatiotemporal derivative filter is a symmetrically positioned pair of "blobs". The gradient constraint states that the blobs can be positioned (by rotating them around the origin) so that multiplying the Fourier transform of the image sequence (the plane) times the frequency response of the filter (blobs) gives zero.

Formally, we rely on Parseval's theorem and the derivative theorem of the Fourier transform to write the gradient method in the frequency domain. Parseval's theorem states that the sum of the squared values in space-time equals the sum of the power in the frequency domain (up to a scalar):

$$\sum_{x,y,t} |f(x, y, t)|^2 \;=\; c \sum_{\omega_x, \omega_y, \omega_t} |F(\omega_x, \omega_y, \omega_t)|^2. \tag{29}$$

The derivative theorem states that the Fourier transform of the derivative of f equals the Fourier transform of f itself multiplied by an imaginary ramp function of the form $j\omega$. For example,

$$\mathcal{F}[f_x(x, y, t)] \;=\; j\omega_x \mathcal{F}[f(x, y, t)] \;=\; j\omega_x F(\omega_x, \omega_y, \omega_t) \tag{30}$$

Combining Eqs. (19), (29), and (30), gives:

$$\begin{aligned} E(\mathbf{u}_1, \mathbf{u}_2) &= \sum_{x,y,t} |\mathbf{u}_1 f_x(x, y, t) + \mathbf{u}_2 f_y(x, y, t) + f_t(x, y, t)|^2 \qquad (31) \\ &= c \sum_{\omega_x, \omega_y, \omega_t} |\mathbf{u}_1 \omega_x F(\omega_x, \omega_y, \omega_t) + \mathbf{u}_2 \omega_y F(\omega_x, \omega_y, \omega_t) + \omega_t F(\omega_x, \omega_y, \omega_t)|^2 \\ &= c \sum_{\omega_x, \omega_y, \omega_t} [(\mathbf{u}_1, \mathbf{u}_2, 1) \cdot \vec{\omega}]^2\, |F(\vec{\omega})|^2. \end{aligned}$$

If all of the assumptions and approximations hold precisely (notably, that the image is uniformly translating for all space and time), then this expression can be precisely minimized by choosing $\vec{\mathbf{u}}$ to satisfy: $\mathbf{u}_1\omega_x + \mathbf{u}_2\omega_y + \omega_t = 0$. If any of the approximations are violated then there will not in general be any $\vec{\mathbf{u}}$ that will satisfy the constraint perfectly. Minimizing Eq. (31) corresponds to a least-squares regression solution for the optical flow.

To be concrete, let $\vec{\omega}_n = (\omega_{xn}, \omega_{yn}, \omega_{tn})$ for $1 < n < N$ be the spatiotemporal frequencies for which we have samples of $|F(\vec{\omega})|$. Rewriting Eq. (31) in matrix notation gives:

$$E(\mathbf{u}_1, \mathbf{u}_2) \;=\; c\,\|\mathbf{A}\vec{u}\|^2, \tag{32}$$

where $\vec{u} = (\mathbf{u}_1, \mathbf{u}_2, 1)^T$ and

$$\mathbf{A} = \begin{bmatrix} F(\vec{\omega}_1)\omega_{x1} & F(\vec{\omega}_1)\omega_{y1} & F(\vec{\omega}_1)\omega_{t1} \\ F(\vec{\omega}_2)\omega_{x2} & F(\vec{\omega}_2)\omega_{y2} & F(\vec{\omega}_2)\omega_{t2} \\ \vdots & \vdots & \vdots \\ F(\vec{\omega}_N)\omega_{xN} & F(\vec{\omega}_N)\omega_{yN} & F(\vec{\omega}_N)\omega_{tN} \end{bmatrix}$$

Because of the weights, $|F(\vec{\omega})|$, the regression depends most on those frequency components where the power is concentrated. Equation (32) is minimized by a vector $\vec{v}$ in the direction of the eigenvector corresponding to the smallest eigenvalue of $\mathbf{A}^T\mathbf{A}$. The image velocity estimate $\hat{\vec{u}}$ is obtained by normalizing $\vec{v}$ so that its third element is 1.

For real image sequences, the least-squares estimate of image velocity (21) is definitely preferred over the frequency domain regression estimate (32) because the frequency domain estimate requires computing a global Fourier transform. But for an image sequence containing a uniformly translating pattern, the two estimates are the same.

4 Multi-Scale, Iterative Motion Estimation

Temporal Aliasing. The image sequences we are given as input often have temporal sampling rates that are slower than that required by the sampling theorem to uniquely represent the high temporal frequencies in the signal. As a consequence, temporal aliasing is a common problem in computing image velocity in real image sequences. The classic example of temporal aliasing is the wagon wheel effect in old Western movies; when the wheel rotates at the right speed relative to the frame rate, its direction of rotation appears to reverse. This effect can most easily be understood by considering the closest match for a given spoke as the wheel rotates from one frame to the next. Let the angular spacing between spokes be α. If the wheel rotates by more than $\alpha/2$ but less than α from one frame to the next, then the wheel will appear to rotate backwards. In fact, any integer multiple of this range will also do so that rotations between $k\,\alpha/2$ and $k\,\alpha$ will appear to rotate backwards (for any integer k).

Temporal aliasing can be a problem for any type of motion algorithm. For feature-based matching algorithms, temporal aliasing gives rise to false matches. For filtering (and gradient-based) methods, temporal aliasing becomes a problem whenever the motion is large compared to the spatial extent of the filters.

The effect of temporal aliasing may also be characterized in the spatiotemporal frequency domain, as illustrated in Fig. 9. Consider a one-dimensional signal moving at a constant velocity. As mentioned above, the power spectrum of this signal lies on a line through the origin. Temporal

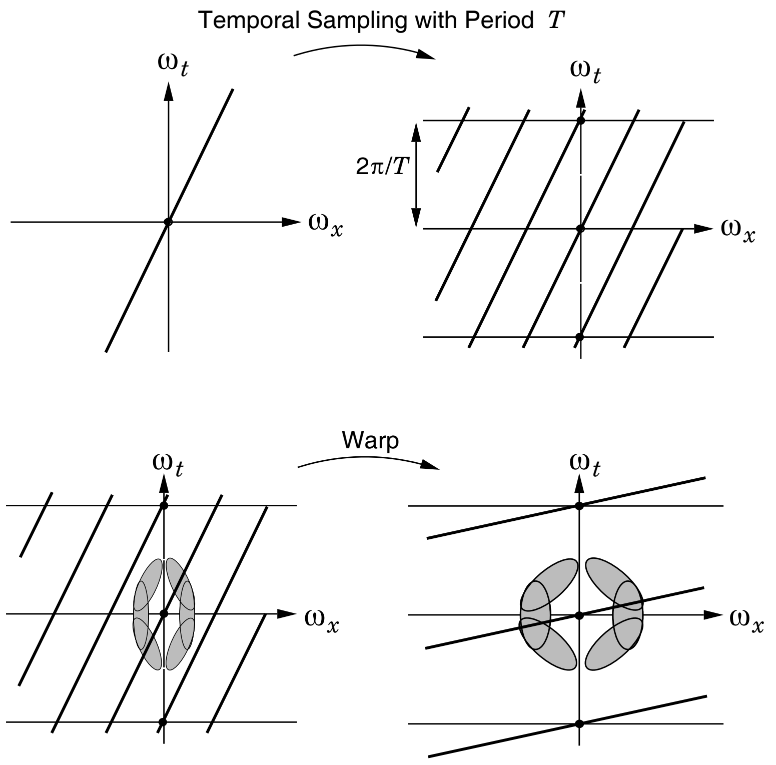


Figure 9: **Top-left:** The power spectrum of a translating signal occupies a line in the spatiotemporal frequency domain. **Top-right:** Sampling in time (to get the discrete frames of an image sequence) gives rise to replicas of the spectra, that may result in temporal aliasing. These replicas can cause severe problems for a gradient-based motion algorithm because the derivative filters may respond as much (or more) to the replicated spectra as they do to the original. **Bottom-left:** The problem can be alleviated by using coarse-scale derivative filters tuned for lower frequencies, i.e., by blurring the images a lot before computing derivatives. The frequency responses of these coarse-scale filters avoid the replicas. **Bottom-right:** The velocity estimates from the coarse scale are used to warp the images, thereby undoing most of the motion. The finer scale filters can now be used to estimate the residual motion; since the velocities (after warping) are slower, the frequency responses of the fine scale filters may now avoid the replicas.

---

sampling causes a replication of the signal spectrum at temporal frequency intervals of $2\pi/T$ radians, where T is the time between frames. It is easy to see that these replicas could confuse a gradient-based motion estimation algorithm because the derivative filters may respond as much (or more) to the replicated spectra as they do to the original

Eliminating Temporal Aliasing by Coarse-to-Fine Warping. An important observation concerning this type of aliasing is that it usually affects only the higher spatial frequencies of an image. In particular, for a fixed velocity, those spatial frequencies moving more than half of their period per frame will be aliased, but lower spatial frequencies will not.

This suggests a simple but effective approach for avoiding the problem of temporal aliasing. Estimate the velocity using coarse-scale filters (i.e., spatially blur the individual frames of the sequence before applying the derivative filters) to avoid the potentially aliased high spatial frequencies. These velocity estimates will correspond to the average velocity over a large region of the image because the coarse-scale filters are large in spatial extent. Given a uniformly translating image sequence, we could stop here. But in typical image sequences, the velocity varies from one point to the next. The coarse-scale filters miss much of this spatial variation in the flow field. To get better estimates of local velocity, we need finer scale (higher frequency) filters with smaller spatial extents.

Towards this, following Bergen *et al.* (1992) for example, the coarse motion estimates are used to warp the images and "undo" the motion, roughly *stabilizing* the objects and features in the images. Then finer scale filters are used to estimate the residual motion; since the velocities (after warping) are slower, the frequency responses of the fine scale filters may now avoid the temporally aliased replicas (as illustrated in Fig. 9C). Typically, such *coarse-to-fine methods* use a an image pyramid data structure to obtain the different low-pass filtered version of the images.

Coarse-to-fine methods are widely used to measure motion in video image sequences because of the aliasing that regularly occurs. Despite this, it does have its drawbacks, stemming from the fact that the fine-scale estimates can only be as reliable as their coarse-scale precursors. For example, if there is little or no signal energy at coarse-scales, then coarse-scale estimates cannot be trusted and we have no initial guess to offer finer-scales. A related problem occurs if the coarse-scale estimate is very poor, perhaps due to the broad spatial extent of the estimation, then the warping will not significantly decrease the effective velocity as desired, in which case the finer-scale estimation will produce an equally poor result.

Iterative Refinement of Velocity Estimates. Such coarse-to-fine estimation strategies are an example of iteratively refining and updating velocity measurements. The process involves warping the images according to the current velocity estimate and then re-estimating the velocity from the warped sequence. The warping is an attempt to remove the motion and therefore *stabilize* the image sequence. This is a useful idea within a coarse-to-fine strategy to avoid the adverse effects of temporal aliasing as mentioned above.

It can also be used to refine the flow estimates without necessarily changing scales. Remember that derivative estimates are often noisy, and the assumptions upon which the velocity estimates were based do not hold exactly. In particular, let's return to the derivation of the gradient constraint in the 1d case, shown in Fig. 4, where we wanted to find the displacement d between $f_2(x)$ and $f_1(x)$. We linearized the signals to derive an estimate $\hat{d} = (f_1(x) - f_2(x))/f_1'(x)$. As illustrated in Fig. 4, when f_1 is a linear signal, then $d = \hat{d}$. Otherwise, to leading order, the accuracy of the estimate is bounded by the magnitude of the displacement and the second derivative of f_1:

$$|\hat{d} - d| \;\leq\; \frac{d^2\,|f_1''(x)|}{2\,|f_1'(x)|}\;. \tag{33}$$

For a sufficiently small displacement, and bounded f_1''/f_1', we expect reasonably accurate estimates.

In any case, so long as the estimation error is smaller than $|d|$, then with Newton's method we can iterate the estimation to achieve our goal of having the estimate $\hat{d}$ satisfy $f_1(x-\hat{d}) - f_2(x) = 0$. Toward this end, assume we're given an estimate $\hat{d}_0$, and let the error be denoted $u = d - \hat{d}_0$. Then, because $f_2(x)$ is a displaced version of $f_1(x)$ we know that $f_2(x) = f_1(x - \hat{d}_0 - u)$. Accordingly, using gradient-based estimation, Eq. (12), we form a new estimate of d given by $\hat{d}_1 = \hat{d}_0 + \hat{u}$, where

$$\hat{u} \;=\; \frac{f_1(x-\hat{d}) - f_2(x)}{f_1'(x-\hat{d})}\;. \tag{34}$$

Here, $f_1(x-\hat{d})$ is simply a warped version of $f_1(x)$. This process can be iterated to find a succession of estimates $\hat{d}_k$ that converge to d if the initial estimate is in the correct direction.

In the general 2D problem, we are given an estimate of the optical flow field $\vec{\mathbf{u}}^0 = (\mathbf{u}_1^0, \mathbf{u}_2^0)^T$ (e.g., it might have been estimated from the coarse scale), and we create a warped image sequence $h(x,y,t)$:

$$h(x,y,\Delta t) = \mathcal{W}[f(x,y,t+\Delta t), \vec{\mathbf{u}}^0(x,y)] = f(x-\mathbf{u}_1^0\Delta t, y-\mathbf{u}_2^0\Delta t, t+\Delta t),$$

where Δt is the time between two consecutive frames, and $\mathcal{W}[f, \vec{\mathbf{u}}^0]$ denotes the warping operation that warps image f by the vector field $\vec{\mathbf{u}}^0$. Note that warping need only be done only over a range of Δt that are required for the temporal differentiation filter.

The warped image sequence $h(x,y,t)$ is then used to estimate the *residual flow field*:

$$\Delta\vec{\mathbf{u}} = \mathbf{M}^{-1}\,\vec{b}$$

where $\mathbf{M}$ and $\vec{b}$ are computed by taking derivatives (via convolution as above) of $h(x,y,t)$. Given the residual flow estimates $\Delta\vec{\mathbf{u}}$, the refined velocity estimates become

$$\vec{\mathbf{u}}^{\,1}(x,y) = \vec{\mathbf{u}}^{\,0}(x,y) + \Delta\vec{\mathbf{u}}(x,y).$$

This new flow estimate is then used to rewarp the original sequence, and the residual flow may be estimated again. In the coarse-to-fine process this new estimate of the flow field is used to warp the images at the next finer scale. This process continues until the finest scale is reached and the residual flow is sufficiently small.

Although we will not work through all the mathematical details, one can also motivate this approach in a more rigorous manner. The estimation described above proceeds in an iterative fashion, with a series of optimizations that converge to a least-squares solution of the desired objective function, i.e.,

$$E(\mathbf{u}) = \sum_{xy} g(x,y)\,[f(x,y,t) - f(x+\mathbf{u}_1^0+\Delta\mathbf{u}_1,\, y+\mathbf{u}_2^0+\Delta\mathbf{u}_2,\, t+1)]^2\ . \qquad (35)$$

In particular, we are linearizing the image difference in (35), with respect to $\Delta\mathbf{u}_1$ and $\Delta\mathbf{u}_2$, to form an approximate objective function which we then solve to find an approximate flow field. You can show that this approximate objective function converges to the desired objective function in (35). In particular, as implied above in (33) the difference between the two objective functions is second-order in the magnitude of the residual flow, $\Delta\vec{\mathbf{u}}_0$. Hence the rewarping the images toward one another with the current estimate of the motion should reduce the residual flow at each iteration so that the approximate solutions converge toward the desired flow (i.e., a locally optimal flow estimate).

5 Higher-Order Gradient Constraints

There are several straightforward ways in which the gradient-based approach introduced above can be extended. One is to introduce other forms of image preprocessing before applying the gradient constraint. If such processing is a derivative filter, then this amounts to deriving high-order differential constraints on the optical flow.

In the gradient-based approach discussed above, we tracked points of consistent intensity (or smoothed version of image intensity) using only first-order derivatives. One can, however use various forms of prefiltering before applying the gradient constraint. For example, the original image sequence might be filtered to enhance image structure that is of particular interest, or to suppress structure that one wishes to ignore. One may also do this to obtain further constraints on the optical flow at a single pixel.

For example, one could prefilter the image with first derivative filters g_x and g_y. Following Nagel et al (1987) and Verri et al (1990), the gradient constraint could then be applied to f_x and f_y (instead of f in as was done above in Eq. (13)). This yields two constraints on the optical flow at each location:

$$\begin{aligned} \mathbf{u}_1 f_{xx} + \mathbf{u}_2 f_{xy} + f_{xt} &= 0 \\ \mathbf{u}_1 f_{xy} + \mathbf{u}_2 f_{yy} + f_{yt} &= 0, \end{aligned} \tag{36}$$

Alternatively, you might use second-order Gaussian derivatives g_{xx}, g_{xy} and g_{yy} as prefilters. In this case you obtain three gradient constraints at each pixel:

$$\begin{aligned} \mathbf{u}_1 f_{xxx} + \mathbf{u}_2 f_{xxy} + f_{xxt} &= 0 \\ \mathbf{u}_1 f_{xxy} + \mathbf{u}_2 f_{xyy} + f_{xyt} &= 0 \\ \mathbf{u}_1 f_{xyy} + \mathbf{u}_2 f_{yyy} + f_{yyt} &= 0, \end{aligned} \tag{37}$$

where f_{xxx} should be interpreted as $f * g_{xxx}$, and likewise for the other derivatives. Equation (37), written in terms of third derivatives, gives three constraints on velocity.

An advantage of using higher order derivatives is that, in principle, there are more constraints at a single spatial position. Even so, it is typically worthwhile combining constraints over a local spatial region and computing a least-squares estimate of velocity by minimizing:

$$\begin{aligned} E(\mathbf{u}_1, \mathbf{u}_2) &= \sum_{x,y} [\mathbf{u}_1 f_{xxx} + \mathbf{u}_2 f_{xxy} + f_{xxt}]^2 \\ &+ \sum_{x,y} [\mathbf{u}_1 f_{xxy} + \mathbf{u}_2 f_{xyy} + f_{xyt}]^2 \\ &+ \sum_{x,y} [\mathbf{u}_1 f_{xyy} + \mathbf{u}_2 f_{yyy} + f_{yyt}]^2 \end{aligned}$$

The solutions can be written in the same form as Eq. (21). The only differences are: (1) that the third derivative formulation uses a greater number of linear filters, and (2) that the third derivative filters are more narrowly tuned for spatiotemporal orientation.

The intensity gradient constraint in Eq. (13) is based on the intensity conservation assumption; it assumes that intensity is conserved (only shifting from one location to another location) over time. The third derivative constraints, Eq. (37), are based on conservation of the second spatial derivatives of intensity, i.e., that f_{xx}, f_{xy}, and f_{yy} are conserved over time.

It is worth noting that one can also view these higher-order constraints in terms of the conservation of some image property. For example, following Eqs. (15) and (17), second-order constraints arise from an assumption that the spatial gradient $\vec{\nabla} f(x, y, t)$ is conserved. In effect, this assumption is inconsistent with dilating and rotating deformations. By comparison, the assumption of intensity conservation in Eq. (16) implies only that image must smoothly deform from one frame to the next, but the form of the flow field is otherwise unrestricted.

6 Higher-Order Subspace Motion Models

A second way we can generalize the gradient-based method above is by introducing richer classes of parametric models for the optical flow within the neighborhood over which constraints are combined. In the method introduced above, we essentially compute a single estimate of the optical flow in the neighborhood. This amounts to an assumption that the flow is well modelled with a constant flow model within the region. It is a relatively straightforward task to generalize this approach to include much more general types of models such as an affine model that allows us to estimate rotation and scaling along with translation in the neighborhood.

So while the multiscale gradient/derivative methods described above compute translational velocity estimates separately in the neighborhood of each pixel, we may wish to estimate more complex parametric models over much larger image neighborhoods. For example, camera motion with respect to a planar surface results in a quadratic flow field. If the entire image happens to correspond to a planar (or nearly planar) surface then you can take advantage of this information. Even if the image does not correspond to a planar surface, a model of the flow field may be a reasonable approximation in local neighborhoods.

Affine Model. Many researchers have found that affine models generally produce much better velocity estimates than constant models (e.g., Fleet and Jepson, 1990; Bergen et al., 1992). An affine motion model is expressed component-wise as

$$\begin{aligned} \mathbf{u}_1(x,y) &= p_1 + p_2 x + p_3 y \\ \mathbf{u}_2(x,y) &= p_4 + p_5 x + p_6 y \end{aligned}$$

where $\vec{p} = (p_1, p_2, p_3, p_4, p_5, p_6)^T$ are the parameters to be estimated. This can be written in matrix notation as:

$$\vec{\mathbf{u}}(x,y) = \mathbf{A}(x,y)\,\vec{p} \tag{38}$$

where

$$\mathbf{A}(x,y) = \begin{bmatrix} 1 & x & y & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & x & y \end{bmatrix}$$

In this instance, the affine model describes the flow field about an origin at location (0,0). If one wished to consider an affine model about the point $(x_0, y_0)^T$ then one might use

$$\mathbf{A}(x,y) = \begin{bmatrix} 1 & x-x_0 & y-y_0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & x-x_0 & y-y_0 \end{bmatrix}.$$

Either way, the same flow fields can be represented using the form of Eq. (38).

The gradient constraint may be written:

$$\vec{f_s}^T \vec{\mathbf{u}} + f_t = 0, \tag{39}$$

where $\vec{f_s} = (f_x, f_y)^T$ are the spatial derivatives. Combining Eqs. (38) and 39 gives:

$$\vec{f_s}^T \mathbf{A}\vec{p} + f_t = 0,$$

so we want to minimize:

$$E(\vec{p}) = \sum_{x,y} (\vec{f_s}^T \mathbf{A}\vec{p} + f_t)^2, \tag{40}$$

where the summation may be taken over the entire image (or over some reasonably large image region). Equation (40) is quadratic in $\vec{p}$ so the estimate $\hat{\vec{p}}$ is derived in the usual (linear least-squares) way by taking derivatives of E with respect to $\vec{p}$. This yields the solution

$$\hat{\vec{p}} = \mathbf{R}^{-1}\,\vec{s}, \tag{41}$$

where

$$\mathbf{R} = \sum \mathbf{A}^T \vec{f}_s \vec{f}_s^T \mathbf{A},$$
$$\vec{s} = -\sum \mathbf{A}^T \vec{f}_s f_t,$$

and where $\sum$ here means that each element of $\mathbf{R}$ and $\vec{s}$ is obtained by summing (element-by-element) over the entire image region. When $\mathbf{R}$ is singular (or ill-conditioned) there is not sufficient image structure to estimate all six of the flow field parameters. This is another instance of the aperture problem discussed above.

Again, multiscale refinement of the parameter estimates is needed to overcome temporal aliasing when the motion is large. We use the following notation:

$$\vec{\mathbf{u}} = \mathbf{A}\vec{p} \quad \vec{\mathbf{u}}^0 = \mathbf{A}\vec{p}^0 \quad \Delta\vec{\mathbf{u}} = \mathbf{A}\Delta\vec{p},$$

so that

$$\Delta\vec{\mathbf{u}} = \mathbf{A}(\vec{p} - \vec{p}^0). \tag{42}$$

As before, we let h denote the warped image sequence and we use its derivatives in the gradient constraint:

$$\vec{h}_s \cdot \Delta\vec{\mathbf{u}} + h_t = 0.$$

Combining with Eq. (42) gives:

$$\vec{h}_s^T \mathbf{A}\vec{p} + (h_t - \vec{h}_s^T \mathbf{A}\vec{p}^0) = 0,$$

so we want to minimize:

$$E(\vec{p}) = \sum [\vec{h}_s^T \mathbf{A}\vec{p} + (h_t - \vec{h}_s^T \mathbf{A}\vec{p}^0)]^2$$

The refined least-squares estimate of the flow field model parameters is therefore:

$$\begin{aligned} \vec{p}^1 &= \mathbf{R}^{-1}(\vec{s} + \mathbf{R}\vec{p}^0) \\ &= \vec{p}^0 + \mathbf{R}^{-1}\vec{s} \end{aligned} \tag{43}$$

where $\mathbf{R}$ and $\vec{s}$ are (as above, except with h instead of f):

$$\mathbf{R} = \sum \mathbf{A}^T \vec{h}_s \vec{h}_s^T \mathbf{A},$$

$$\vec{s} = -\sum \mathbf{A}^T \vec{h}_s h_t,$$

In other words, the refinement of the model parameters has the same form as the refinement of the flow fields estimates above. The residual model parameters $\Delta\vec{p}$ are given by

$$\Delta\vec{p} = \mathbf{R}^{-1}\vec{s}$$

Similarity Deformation. Sometimes it is natural to express the expected deformation as a similarity transformation, comprising translation, rotation and uniform scaling. In this case, the flow in the neighbourhood of the point $(x_0, y_0)^T$ can be expressed as

$$\vec{\mathbf{u}}(x,y) \;=\; a \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{pmatrix} x - x_0 \\ y - y_0 \end{pmatrix} \;+\; \vec{\mathbf{d}} \tag{44}$$

for translation $\vec{\mathbf{d}}$, rotation angle θ, and scaling factor a. We can express this deformation in a linear basis of the form

$$\vec{\mathbf{u}}(x,y) = \mathbf{A}(x,y)\,\vec{p} \tag{45}$$

where

$$\mathbf{A}(x,y) = \begin{bmatrix} 1 & 0 & x - x_0 & -y + y_0 \\ 0 & 1 & y - y_0 & x - x_0 \end{bmatrix}$$

and $\vec{p} = (\vec{\mathbf{d}}^T, a\cos\theta, a\sin\theta)^T$. With this linear form, we can then solve directly for the parameter vector $\vec{p}$ using the same approach described above, from which we can then compute the angle and scale if need be.

Rigid, Quadratic and Perspective Deformations. There are many other transformations that are commonly used in motion estimation. Three common deformations are the rigid deformation, the quadratic deformation and plane perspective deformation (Bergen et al, 1992).

The three parameter 2D rigid transform in the neighbourhood of a point $(x_0, y_0)^T$,

$$\vec{\mathbf{u}}(x,y) \;=\; \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{pmatrix} x - x_0 \\ y - y_0 \end{pmatrix} \;+\; \vec{\mathbf{d}}\,. \tag{46}$$

This is particularly useful in image processing applications such as reducing the effects of camera jitter, for which one knows the image deformation is approximately rigid. In order to enforce the constraint that the matrix be orthogonal, the estimation of the parameters of rigid motion models is somewhat more complex than the linear estimation outlined above.

Quadratic deformations are relatively easy to estimate because they can be expressed with a linear subspace, just like the affine and similarity deformations. They are special because, as shown already in Section (1) with Eq. (9), they are produced from the instantaneous motion of a planar surface under perspective projection

Finally, a perspective deformation of points on a plane in 3D is an 8-parameter linear deformation in homogeneous coordinates, known as a homography. It maps 2D image points at one time (expressed in homogeneous coordinates as 3-vectors) to points at the next time. This deformation is important since it exactly captures the image motion caused by the rigid 3D motion of points on a plane between two discrete time instants.

Learned Subspace Models. Finally, it is important to remember that not all flow fields are well expressed as low-dimensional polynomial deformations. In general they are neither that smooth, nor that generic. Many objects have very distinct, characteristic patterns of motion, some examples of which include human mouths while people are speaking, people's legs while walking, or even the motion at occlusion boundaries where the motion is neither smooth nor continuous. In cases likes these it is sometimes useful to construct domain-specific motion models. As above, it is convenient to express these models using parametric linear subspaces. However the basis flow fields will not be simple constant, linear or quadratic functions of spatial position. Bregler and Malik (1998) take this approach in their work on estimating the flow of articulated objects. Fleet et al. (2000) used a subspace approach based on steerable bases to construct a motion model for estimating optical flow in the neighborhoods of occlusion boundaries. They also show how parameterized linear motion models can be learned from examplars in natural image sequences. This approach was applied successfully to people speaking and walking.

7 Phase-Based Methods

The gradient constraint is useful for tracking any differentiable image property. We can apply it to properties other than the image intensities or linearly filter versions of the image. One could apply various forms of nonlinear preprocessing instead.

There are two approximations underlying the gradient constraint (see above): (i) the image intensity conservation assumption, and (ii) the numerical approximation of first-order partial derivatives in space and time. With raw image intensities the intensity conservation assumption is often violated, and derivatives can be difficult to estimate reliably. However, with appropriate preprocessing of the image sequence one may satisfy the conservation approximation more closely and also improve the accuracy of numerical differentiation. For example, local gain control can compensate for changes in illumination and thereby make a significant difference.

One successful example of nonlinear preprocessing, proposed by Fleet and Jepson (1990, 1993), is to use the phase output of quadrature-pair linear filters. Phase-based methods are a combination of frequency-based and gradient-based approaches; they use band-pass prefilters along with gradient constraints and a local affine model to estimate optical flow.

Phase Correlation. To explain phase-based approaches it is useful to begin with a method called phase-correlation (Kuglin and Hines, 1975). The method is often derived by assuming pure translation between two signals (here in the 1d case for simplicity):

$$f_2(x) = f_1(x-d) \ . \tag{47}$$

The Fourier shift theorem tells us that their Fourier transforms satisfy

$$F_2(\omega) = F_1(\omega) e^{-i\,d\omega} \ . \tag{48}$$

Their amplitude spectra are identical, $A_1(\omega) = A_2(\omega)$, and their phase spectra differ by $d\omega$, i.e., $\psi_2(\omega) = \psi_1(\omega) - d\omega$.

Taking the product of $F_1(\omega)$ and the complex conjugate of $F_2(\omega)$, and then dividing by the product of their amplitude spectra, we obtain

$$\frac{F_1(\omega)\,F_2^*(\omega)}{A_1(\omega)A_2(\omega)} = \frac{A_1(\omega)A_2(\omega)e^{i(\psi_1(\omega)-\psi_2(\omega))}}{A_1(\omega)A_2(\omega)} = e^{i\omega d} \tag{49}$$

Here, $e^{i\omega d}$ is a sinusoidal function in the frequency domain, and therefore its inverse Fourier transform is an impulse, located at the displacement d, that is, $\delta(x+d)$. Thus, in short, phase-correlation methods measure displacement by finding peaks in

$$\mathcal{F}^{-1}\left[\frac{F_1(\omega)\,F_2^*(\omega)}{A_1(\omega)A_2(\omega)}\right] \tag{50}$$

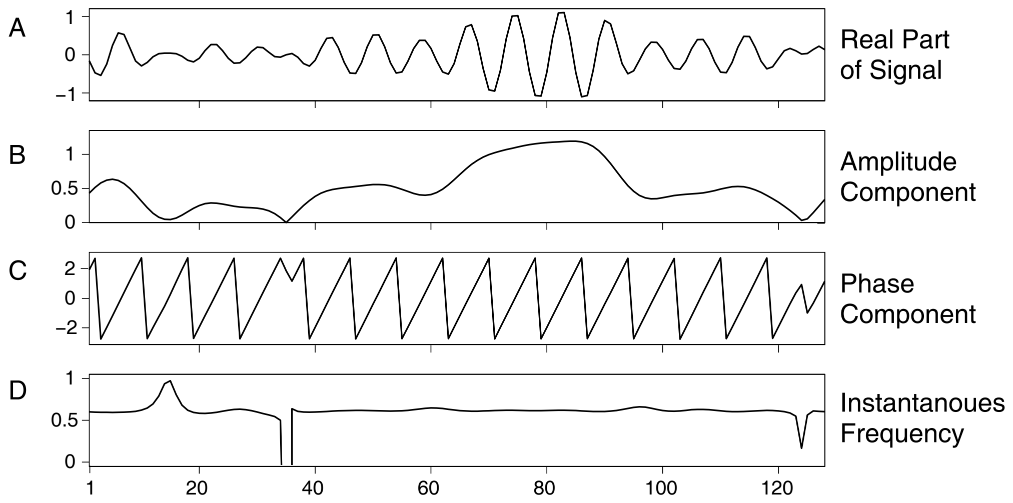


Figure 10: A general band-pass signal can be described in terms of an amplitude component and a phase component. The instantaneous frequency is the derivative of phase with respect to spatial position, and is usually close to the signals pass-band frequencies.

Phase correlation, by using a global Fourier basis, measures a single displacement for the entire image. But for many applications of interest, we really want to measure image velocity locally. So, a natural alternative to the Fourier basis is to use a wavelet basis. This gives us a local, self-similar representation of the image from which we can still use phase information (Fleet, 1994).

Local Phase Differences. To develop local phase-based methods, we first need to introduce some notation and terminology associated with band-pass signals. First, with 1d signals and filters, let's assume that we have a band-pass filter $h(x)$ and its response $r(x)$, perhaps from a wavelet transform. Let's also assume that $r(x)$ is over-sampled to avoid aliasing in individual subbands.

Because $r(x)$ is band-pass, we expect it to modulate at frequencies similar to those in the passband. One model of a band-pass signal is given by $r(x) = a(x)\cos(\phi(x))$, where $a(x)$ and $\phi(x)$ are referred to as the amplitude and phase components of the signal. The amplitude component describes the amplitude of local modulation. The phase component describes the local structure of the signal; e.g., $\phi(x_0) = \pm\pi/2$ implies that the signal has a zero-crossing at x_0, and $\phi(x_0) = 0$ implies that the signal has a peak near x_0.

In addition to phase and amplitude, another useful quantity is the instantaneous frequency of $r(x)$ at a point x, defined as $k(x) = d\,\phi(x)/dx$. To understand this definition intuitively, remember that a sinusoidal signal with constant amplitude is given by $\sin(\omega x)$, in which case the phase component is $\phi(x) = \omega\, x$ and the instantaneous frequency is $k(x) = \omega$. With general band-pass signals, unlike this sinusoid, the instantaneous frequency will vary as a function of spatial position.

Figure 10 shows a band-pass signal, along with its amplitude and phase signals, and its instantaneous frequency. Figure 10A shows the real part of the signal, as a function of spatial position, while Figures 10B–C show the amplitude and phase components of the signal. Unlike Fourier spectra, these two signals are functions of spatial position and not frequency. The amplitude represents the magnitude of the signal modulation. The phase component represents the local structure of the signal (e.g., whether the local structure is a peak, or a zero-crossing). Figure 10D shows the instantaneous frequency.

Normally, as can be seen in Figure 10, amplitude is slowly varying, the phase is close to linear, and the instantaneous frequency is close to the center of the pass-band of the signal's Fourier spectrum (about 0.6 in Figure 10D). This characterization suggests a crude but effective model for a band-pass signal about a point x_0, i.e.,

$$r(x) \;\approx\; a(x_0)\,\cos(\phi(x_0) \,+\, (x - x_0)\phi'(x_0)) \;. \tag{51}$$

This approximation involves a constant approximation to the amplitude signal and a first-order approximation to the phase component.

With respect to estimating displacements, we can now point to some of the interesting properties of phase. First, if the signal translates from one frame to the next, then, as long as the filter is shift-invariant, both the amplitude and phase components will also translate. Second, phase almost never has a zero gradient, and it is linear over relatively large spatial extents compared to the signal itself. These properties are useful since gradient-based techniques require division by the gradient, and the accuracy of gradient-based measurements is a function of the linearity of the signal. Together, all this is promising for phase-gradient methods.

Therefore, given two phase signals, $\phi_1(x)$ and $\phi_2(x)$, we can estimate the displacement required to match phase values at a position x using gradient constraints. Fleet et al. (1991) proposed a constraint of the form

$$\hat{d} \;=\; \frac{\Delta\phi(x)}{\tilde{\omega}(x)} \tag{52}$$

where $\tilde{\omega}(x)$ is a measure of the local instantaneous frequency, such as $\tilde{\omega}(x) = (\phi_1'(x) + \phi_2'(x))/2$, and $\Delta\phi$ denotes the phase difference between the two signals at a position x. Because phase is defined uniquely only between $-\pi$ and π, we need to compute the phase difference $\phi_1(x) - \phi_2(x)$ modulo 2π. As with intensity gradient-based methods, this estimator can be iterated using a form of Newton's method to obtain a final estimate. However, in many cases the estimator is accurate enough that only one or two iterations are necessary.

2D Phase-Based Flow and Practical Issues. This method generalizes straightforwardly to 2D, yielding one phase-based gradient constraint at every pixel for every filter. A natural class of filters would be a steerable pyramid, the filters of which are orientation- and scale-tuned.

With a steerable pyramid, the easiest way to extract phase and instantaneous frequency is to use quadrature-pair filters (Adelson and Bergen, 1985; Freeman and Adelson, 1991). These a complex-valued filters the real and imaginary parts of which are Hilbert transforms of one another. In the Fourier domain, they occupy space in only one half-plane. And because the filter are complex-valued (a pair of real-valued filters), the output $r(x, y, t)$ is also complex-valued. Accordingly, the amplitude components of the output is just the square-root of the sum of the squares of the real and imaginary outputs, i.e. $|r(x)|$. The phase component is the phase angle of $r(x)$, often written as $\arg(r(x))$, and computed using `atan2`.

In order to compute the phase difference, it is convenient to use

$$\lfloor \phi_1(x) - \phi_2(x) \rfloor \;=\; \arg(r_1(x)\, r_2^*(x)) \;. \tag{53}$$

And to compute the phase gradient it is convenient to use the identity

$$\frac{d\phi(x)}{dx} = \frac{\mathrm{Im}[r'(x)r^*(x)]}{|r(x)|^2} . \tag{54}$$

This is convenient since, because $r(x)$ is band-pass we know how to interpolate and differentiate it. Conversely, because phase is only unique in the interval $[-\pi, \pi)$, there are phase wrap-arounds from π to $-\pi$ which makes explicit differentiation of phase awkward.

Finally, because the responses $r(x, y, t)$ have two spatial dimensions, we take partial derivatives of phase with respect to both x and y, yielding gradient constraint equations of the form

$$\mathbf{u}_1 \phi_x(x, y, t) + \mathbf{u}_2 \phi_y(x, y, t) + \Delta\phi(x, y, t) = 0 , \tag{55}$$

where $\Delta\phi(x, y, t)$ denotes the phase difference $\phi(x, y, t + 1) - \phi(x, y, t)$ modulo 2π. Since we have many filters in the steerable pyramid, we get one such constraint from each filter at each pixels. We may combine such constraints over all filters within a local neighbourhood and then compute the best least-squares fit to a motion model (e.g. an affine model). And of course this can be iterated to achieve convergence as above.

Also, if it is known that aliasing does not exist in the spatiotemporal signal, then we can also differentiate the response in time, and hence compute the temporal phase derivative. This yields constraints of the form

$$\mathbf{u}_1 \phi_x(x, y, t) + \mathbf{u}_2 \phi_y(x, y, t) + \phi_t(x, y, t) = 0 , \tag{56}$$

And we may use phase constraints at all scales simultaneously to estimate a local model of the optical flow.

Multiscale Phase Differences Of course, as is often the case with conventional image sequences, like those taken with a video camera, there may be aliasing so that one can not differentiate the signal with respect to time. With respect to phase-based techniques, any signal that is displaced more than half a wavelength is aliased, and therefore phase-based estimates of the displacement will be incorrect. Thus, estimates from responses from filters tuned to higher frequencies will have problems at smaller displacements than responses from filters tuned to lower frequency.

For example, Figure 11 shows the consequences of aliasing. The bandwidth of the 4 filters was approximately 2 octaves, and the passbands were centered at frequencies with wavelengths $\lambda = 4, 8, 16$, and 32 pixels. Figure 11A shows the distributions of instantaneous frequency (weighted by amplitude) that one obtains from these filters with the Einstein image. For relatively large amplitudes, the instantaneous frequencies are concentrated in the respective passbands.

So instantaneous frequencies in the response to the filter with $\lambda = 4$ are generally close to $2\pi/4$. According, we expect to see aliasing appear when the displacements approach 2 pixels (half a wavelength). Indeed, Figure 11B shows that the distribution of displacement estimates obtained from the four filters is very accurate when the displacement is very small. By comparison, when the displacement is -2.5 pixels, as in Figure 11C, one can see that the distribution of estimates from the high frequency filter broadens. As expected, given wavelengths near 4 and a displacement of -2.5, the aliased distribution of estimates occurs near a disparity of 1.5 pixels. When the displacement is 4, in Figure 11D, the high frequency filter produces a broad distribution of aliased estimates near 0. The filter tuned to $\lambda = 8$ also begins to show signs of aliasing. Finally, in Figure 11E, with a displacement of -7.5, aliasing causes erroneous estimates is all but the lowest frequency filter.

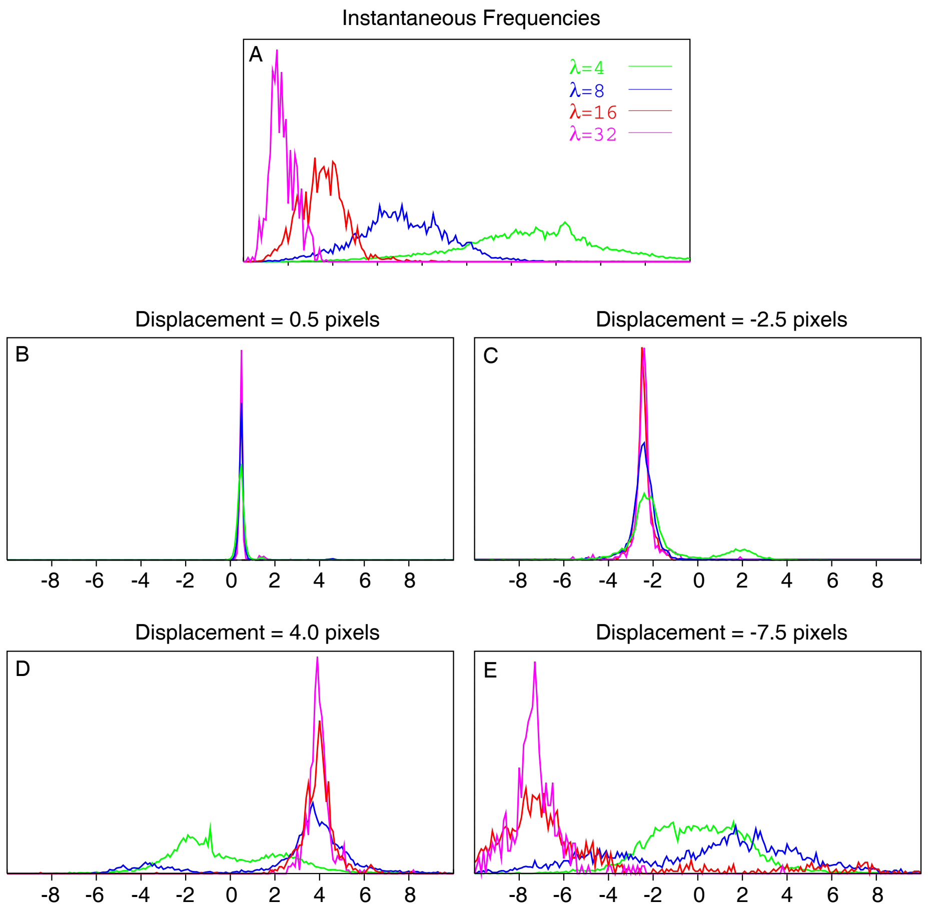


Figure 11: Thanks to Jepson for this figure. Need a caption here.

Clearly, some form of control structure is required so that errors due to aliasing can be detected and ignored. When talking about gradient based methods we introduced a coarse-to-fine control strategy that could be used here as well. The phase correlation methods suggests an alternative approach. Yet another approach, somewhat similar to phase correlation, is to consider evidence through scale at a wide variety of shifts, and choose that displacement at which the evidence at all channels is consistent (Fleet, 1994). This approach is beyond the scope of this handout.

Phase Stability and Singularities. There are several reasons for the success of phase-based techniques for measuring displacements and optical flow. One is that phase signals from band-pass filters are pseudo-linear in local space-time neighborhoods of an image sequence. The extent of the linearity can be shown to depend on the bandwidth of the filters; narrow bandwidth filters produce linear phase behavior over larger regions (Fleet and Jepson, 1993).

Another interesting property of phase is it stability with respect to deformations other than translation between views of a 3D scene. Fleet and Jepson (1991,1993) showed that phase is stable with respect to small affine deformations of the image (e.g., scaling the image intensities and/or adding a constant to the image intensities), so that local phase is more often conserved over time. Again, it turns out that the expected stability of phase through scale depends on the bandwidth

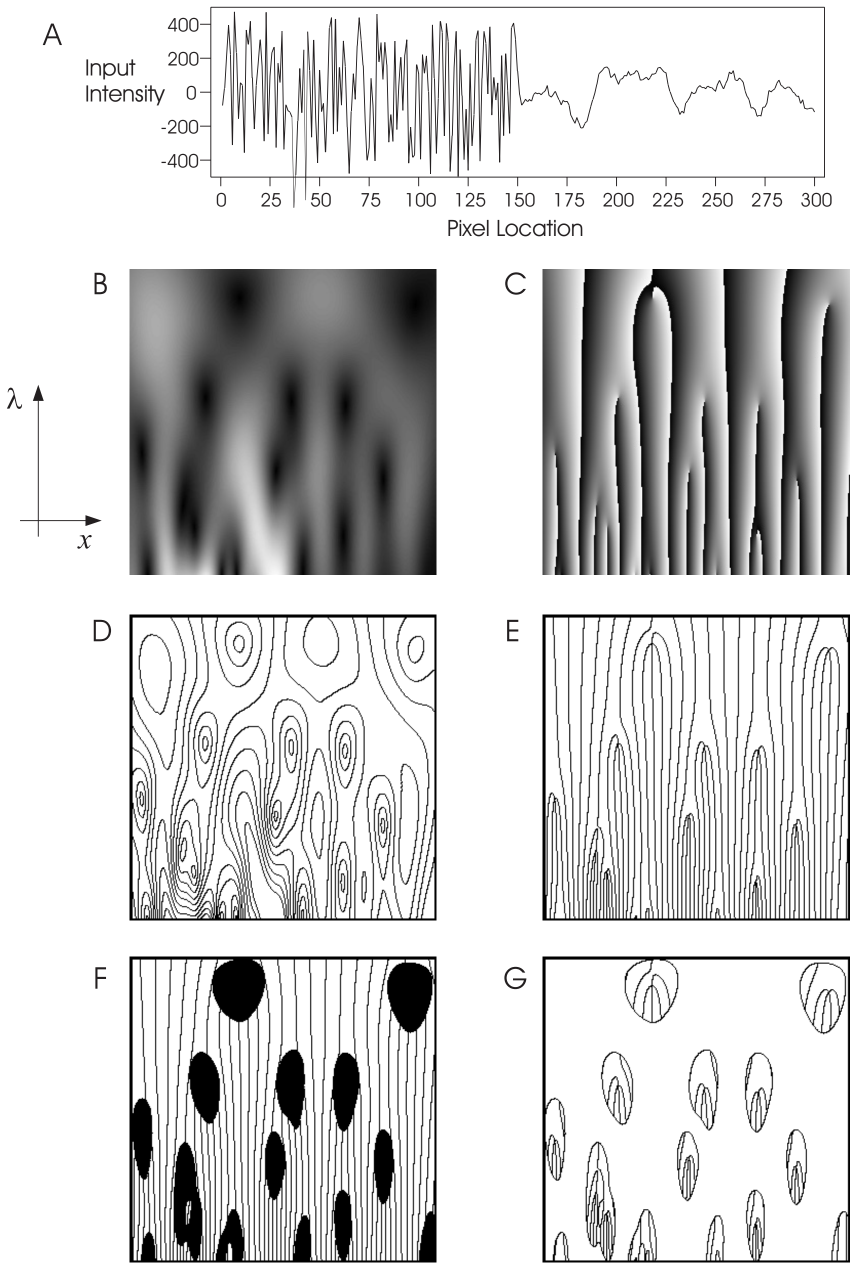


Figure 12: **(A)** An input signal; its left half is a sample of white Gaussian noise, and the right half is a sample of pink Gaussian noise. The amplitude **(B)** and phase **(C)** responses are shown for a complex-valued Gabor filter tuned to wavelength λ. Their iso-amplitude and phase contours are shown in **(D)** and **(E)**. These contours clearly show the relative stability of these two signals. Singularity neighbourhoods (shown in **(G)**) are shown deleted in **(F)**.

---

of the filters. The broader the bandwidth the more stable phase becomes.

For example, Figure 12B,C show the amplitude and phase components of a scale-space expansion $S(x, \lambda)$ of the Gaussian noise signal in Figure 12A. This is the response of a complex-valued Gabor filter with a half-amplitude bandwidth of 0.8 octaves, as a function of spatial position x and wavelength λ, where $16 \leq \lambda \leq 64$ pixels. Level contours of amplitude and phase are shown below in Figure 12D,E. In the context of the scale-space expansion, an image property is said to be *stable* for image matching where its level contours are vertical. In this figure, notice that the amplitude contours are not generally vertical. By contrast, the phase contours are vertical, except for several isolated regions.

These isolated regions of instability are called singularity neighborhoods. In such regions, it can be shown that phase is remarkably unstable, even to small perturbations of scale or of spatial position (Fleet et al, 1991; Fleet and Jepson, 1993). Accordingly, phase constraints on motion estimation should not be trusted in singularity neighborhoods. To detect pixels in singularity neighborhoods, Fleet and Jepson (1993) derived the following reliability condition:

$$\sigma(\lambda) \left| \frac{\partial}{\partial x} \log S(x,\, \lambda) \;-\; i\, k(\lambda) \right| \quad \leq \quad \tau \;, \tag{57}$$

where $k(\lambda) = 2\pi/\lambda$, and $\sigma(\lambda)$ is the standard deviation of the Gaussian envelope of the Gabor filter, which increases with wavelength. As τ decreases, this constraint detects larger neighbourhoods about the singular points. In effect, the left-hand side of Eq. (57) is the inverse scale-space distance to a singular point. For example, Figure 12F shows phase contours that survive the stability constraint, with $\tau = 1.25$, while Figure 12G shows the phase contours in regions removed by the constraint, which amounts to about 20% of the entire scale-space area.

8 Probabilistic Motion Estimation and Robust Regression

The least-squares estimators outlined above provide useful estimates of optical flow, but they do not help derive suitable confidence bounds. Nor do they help one know how to incorporate prior information one might have about optical flow to help constrain the estimation. As a result, one may not be able to propagate flow estimates from one time to the next, nor know how to weight them when combining flow estimates from one source with those from another source.

A different problem with least-squares estimators occurs when one or more of the motion constraints (e.g., the gradient- or phase-based constraints) are clearly inconsistent with the true underlying image motion. This occurs in natural images for a wide variety of reasons, including when time-varying specularities are present (whose motion is quite different from that of the actual object surface), and when flow measurements are taken in the neighborhood of occlusions (where multiple motions exist in a small region). In cases like these the least-squares optical flow estimates will not reflect a single motion. Rather they will be some mixed estimate that can be arbitrarily far from the motion of the surface in which one is interested.

To help cope with these and other limitations of least-squares approaches, various researchers have moved to consider probabilistic formulations.

Probabilistic Least-Squares Formulation. In effect, the simple least-squares solution in (21) corresponds to an optimal solution when one has no prior belief about the true optical flow, and the noise in the gradient constraint is additive, mean-zero Gaussian, and independent in each distinct gradient constraint. In particular, let's assume, for sake of argument that any violation of the gradient constraint (13) can be expressed as noise in the temporal derivative measurement. That is, assume that our measurements of the spatial image gradient (f_x, f_y) are accurate, but our measurement of the temporal image derivative (or image difference) is noisy:

$$\tilde{f}_t \;=\; f_t + \eta \;, \tag{58}$$

where f_t is the true value, $\hat{f}_t$ is the noisy measurement, and η has a Gaussian density function with a mean of zero, a variance of σ^2 at every point, and it is statistically independent at each position and time.

Accordingly, it is easy to show that the likelihood of observing a space-time image gradient measurement (f_x, f_y, f_t), given (or conditioned on) the flow $\vec{\mathbf{u}}$, is therefore

$$p(f_x, f_y, \tilde{f}_t \,|\, \vec{\mathbf{u}}) \;=\; \frac{1}{\sqrt{2\pi}\sigma} \exp\left(\frac{-(f_x \mathbf{u}_1 + f_y \mathbf{u}_2 + \tilde{f}_t)^2}{2\sigma^2} \right) \;. \tag{59}$$

A maximum likelihood (ML) estimator is one that finds the flow $\mathbf{u}$ that maximizes this likelihood of observing the measurements (i.e., 'most consistent' with the measurements). Maximizing the likelihood is equivalent to minimizing the negative log likelihood,

$$-\log p(f_x, f_y, \tilde{f}_t \,|\, \vec{\mathbf{u}}) \;=\; -\log(\sqrt{2\pi}\sigma) \;+\; \frac{(f_x \mathbf{u}_1 + f_y \mathbf{u}_2 + \tilde{f}_t)^2}{2\sigma^2} \;. \tag{60}$$

Finally, if with the assumption that the noise in each measurement is independent, the likelihood of a collection of local gradient measurements is simply the product of the individual likelihoods. Equivalently, the negative log likelihood is the sum of individual log likelihoods

$$\begin{aligned} L(\vec{\mathbf{u}}) \;&=\; -\sum_{x,y} \log p(f_x(x,y,t), f_y(x,y,t), \tilde{f}_t(x,y,t) \,|\, \vec{\mathbf{u}}) \\ &=\; -N \log(\sqrt{2\pi}\sigma) \;+\; \sum_{x,y} \frac{(f_x(x,y,t)\mathbf{u}_1 + f_y(x,y,t)\mathbf{u}_2 + \tilde{f}_t(x,y,t))^2}{2\sigma^2} \;, \end{aligned} \tag{61}$$

where the sum is taken over N positions in the neighborhood. Clearly, if we simply make one more assumption that the inverse variance scales like a Gaussian function with the distance from the location (x, y) to the center of the spatial neighborhood, then the flow that minimizes (61) is the same as that which minimizes the least-squares error in (19) above.

With this formulation we obtain several advantages over the corresponding least-squares formulation given earlier. First, we gain an understanding the noise model that we was implicitly assumed. This is clearly something we might take issue with in deriving other estimators of interest (see below). Second, this formulation also provides a likelihood distribution. As a consequence, it provides information about the relative merits of nearby motions. This is particularly useful for assessing our confidence in the resulting estimate (the Cramer-Rao lower bound is outside the scope of this introduction). Finally, within this framework we can also incorporate prior information about image motions. In that case it is common to maximize the posterior distribution (producing a MAP estimate), which is proportional to the product of the likelihood function and ones prior probability distribution over the domain of possible motions. (e.g., Simoncelli et al. (1991) introduce a prior on slow motions).

Total Least-Squares. While it makes sense that noise in numerical temporal differentiation may be larger than noise in spatial differentiation, it does not make sense that there are no errors in our spatial derivative measurements. Accordingly Weber and Malik (1995) used *total least squares*

(TLS) for estimating flow. This can be shown to be a maximum likelihood error, but only in the case of isotropic errors (Nestares et al, 2000). Arguably this is also a bad assumption. Alternatively one can formulate maximum likelihood estimators where the temporal derivatives have more noise than spatial derivatives (Nestares and Fleet, 2003). Further mathematical details on this topic are beyond the scope of this introduction.

Outliers and Robust Estimation. In many cases, the more significant problem with the simple least-squares estimator is that the errors are not Gaussianly distributed. Rather, it is often the case that a significant fraction of the constraints have errors that are much larger than anything you would expect from a Gaussian model. We say that such an error distribution is a long-tailed distribution.

Cases like this are usually caused by violations of the model assumptions. This include the brightness constancy assumption, and the fact that we only expect a single parametric motion model to account for the deformation within the image region throughout which we gather the motion constraints. Brightness constancy is violated in many cases where the motion of an object causes variations in its radiance (e.g., due to changing diffuse reflectance as its surface orientation changes, specular reflections, or time-varying shadows). The coherence of the local motion is a reasonable assumption for smooth surfaces, but with scene containing a significant amount of depth variation, occlusions become a problem. If an occlusion boundary intersections the image neighborhood over which constraints are collected, then clearly some constraints reflect the motion on one side of the boundary while other constraints reflect the motion on other side.

In cases like this we can use robust regression techniques to effectively ignore such outlying motion constraints. One example of a robust regression for motion estimation approach is that of Black and Anandan (1996).

9 Layered Motion: Mixture Models and EM

When an affine or constant model is a good approximation to the image motion, area-based regression with gradient-based constraints are very effective. One of the main reasons is that, with these methods one estimates only a small number of parameters (6 for the affine flow model) given hundreds or thousands of constraints (one constraint for each pixel throughout a large image region). The problem with such model-based methods is that large image regions are often not well modeled by a single parametric motion due to the complexity of the motion or the presence of multiple moving objects. Smaller regions on the other hand may not provide sufficient constraints for estimating the parameters (the generalized aperture problem discussed above).

One possible solution to this problem is to use relatively large regions of support for the motion estimation, but with a model of motion layers in which different pixels can be assigned to one or more layers, each with its own motion. In effect this is like a cardboard cutout representation of a scene in which different cardboard surfaces correspond to different layers, and they are assumed to be able to move independently [e.g., see (Wang and Adelson, 1994; Black and Jepson, 1996, Ju *et al.*, 1996)].

Among other things this approach can help provide better estimates of motion near locations of occlusion boundaries, or when multiple nearby motions both appear to explain (or fit) the motion constraints in a region similarly well. For example, flow discontinuities occur at depth discontinuities, such as the edge of the can in Figure 13. In this scene the camera was translating,

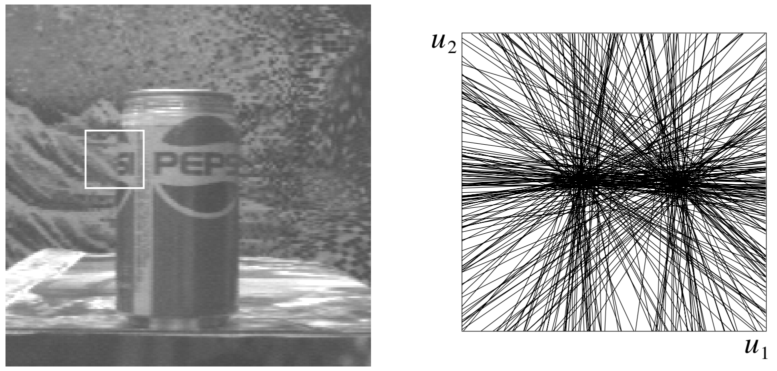


Figure 13: The depth discontinuity at the left side of the can creates a motion discontinuity as the camera moves to the right. Motion constraint lines in velocity space are shown on the right, taken from pixels sampled within the white square. (From Jepson and Black (1993).)

and therefore both the can and the background move with respect to the camera, but with different image velocities. To demonstrate this, let's just consider the gradient constraints in the small region (marked in white) at the left side of the can. Figure 13(right) shows this collection of gradient-constraint lines in velocity space. There appear to be two locations with a high density of constraint-line intersections, and these correspond to the velocities of the can and the background.

A clean formulation of layered motion estimation can be developed using probabilistic mixture models, with the Expectation-Maximization (EM) algorithm used for parameter estimation (Jepson and Black, 1993). The goal of mixture model estimation is to find the motion parameters for each of the individual motions, and to determine which pixels in the region are *owned* by each of the motions.

Mixture Models Let's say that we have an region of pixels $\{\vec{x}_k\}_{k=1}^{K}$ in which we suspect there are multiple velocities present. The region might contain an occlusion boundary for example. By way of notation, let $\vec{\mathbf{u}}(\vec{x};\vec{a})$ denote a parameterized flow field with parameters $\vec{a}$. The parameter vector would be 2-dimensional for a constant model, for example, or 6-dimensional for an affine model. Within a single region of the image we will assume that there are N distinct motions, parameterized by $\vec{a}_n$, for $1 \leq n \leq N$.

Thus, within the region of interest there are N motions, and hence N vectors of unknowns, $\vec{a}_n$. According to the our new *mixture model* the individual motions occur with probability m_n. These *mixing probabilities* tell us what fraction of the K pixels within the region we expect to be consistent with (i.e., *owned* by) each motion. Of course they must all sum to 1.

Let's further assume that we extract one gradient-based motion constraint at each pixel within the region. As above, let $\vec{c}_k$ denote the vector of spatial and temporal image derivatives $\vec{c}_k = (f_x(\vec{x}_k),\, f_y(\vec{x}_k),\, f_t(\vec{x}_k))$ at pixel $\vec{x}_k$ where K is the number of pixels in the region and $1 \leq k \leq K$. As above, we don't expect the gradient constraint to be exactly satisfy the correct motion model. Rather, As a measure of error (inconsistency) let's assume that the dot product

$$d(\vec{\mathbf{u}},\, \vec{c}) \;=\; (\mathbf{u}_1,\, \mathbf{u}_2,\, 1) \cdot \vec{c}$$

is a Gaussian random variable, with a mean of zero and a standard deviation of σ_v. Thus, the likelihood of observing a constraint $\vec{c}$ given the n^{th} flow model, written as $p_n(\vec{c}_k \,|\, \vec{a}_n)$, is $p_n(\vec{c}_k \,|\, \vec{a}_n) = G(d(\vec{\mathbf{u}}_n, \vec{c});\, \sigma_v)$ where $G(d;\, \sigma)$ denotes a mean-zero Gaussian with standard deviation σ, evaluated at d.

Finally, with the mixing probabilities and with the observation densities for the gradient constraints (conditioned on the motion parameters), the mixture model expresses the probability of a single observation (i.e., a constraint) $\vec{c}_k$, as

$$p(\vec{c}_k \,|\, \vec{m},\, \vec{a}_1,\, ...,\, \vec{a}_N) \;=\; \sum_{n=1}^{N} m_n\, p_n(\vec{c}_k \,|\, \vec{a}_n)\ .$$

This says that the probability of observing $\vec{c}_k$ given the motion mixture is a weighted sum of the probabilities of observing $\vec{c}_k$ from the individual motions.

With this model we can write the likelihood of observing a set of K independent constraints $\{\vec{c}_k\}_{k=1}^{K}$ as the product of the marginal probabilities:

$$L(\vec{m},\, \vec{a}_1,\, ...,\, \vec{a}_N) \;=\; \prod_{k=1}^{K} p(\vec{c}_k \,|\, \vec{m},\, \vec{a}_1,\, ...,\, \vec{a}_N)\ .$$

Our goal is to find the mixture model parameters (the mixture proportions and the motion model parameters) that maximize the likelihood of observing the data. Towards this end, given the exponential form of many probability functions, it is common to maximize the log likelihood:

$$\log L(\vec{m},\, \vec{a}_1,\, ...,\, \vec{a}_N) \;=\; \sum_{k=1}^{K} \log p(\vec{c}_k \,|\, \vec{m},\, \vec{a}_1,\, ...,\, \vec{a}_N) \;=\; \sum_{k=1}^{K} \log\left(\sum_{n=1}^{N} m_n\, p_n(\vec{c}_k \,|\, \vec{a}_n)\right)\ .$$

EM and Ownerships. The expectation-minimization (EM) algorthm is a general technique for maximum likelihood or MAP parameter estimation (Dempster et al, 1977; McLachlan and Basford, 1988). The framework is usually explained in terms of a parametric model, some observed data, and some unobserved data. In our case the observed data are the gradient constraints at each pixel, the model parameters are the mixing probabilities and the motion model parameters, and finally the unobserved data are the assignments of gradient constraints to motion models. In particular, note that if we knew which constraints should be associated with which of the N motions, then we could solve for each motion independently from their respective constraints.

Roughly speaking, the EM algorithm is an iterative algorithm that iterates two steps that compute 1) the expected values of the unobserved data given the more recent estimate of the model parameters (the E Step), and then 2) the ML/MAP estimate for the model parameters given the observed data, and the expected values for the unobserved data.

A key quantity in this algorithm is called an *ownership probability.* An ownership probability, denoted $q_n(\vec{c}_k)$, is the probability that the n^{th} motion model is responsible for the constraint (i.e., generated the oberved data) at pixel $\vec{x}_k$. This is an important quantity as it effectively segments the region, telling us which pixels (i.e., data constraints) belong to which motion model.

Using Bayes' rule, we can write probability that data d_k is owned by model $\mathcal{M}_n$ as

$$p(\mathcal{M}_n \,|\, d_k) \;=\; \frac{p(d_k \,|\, \mathcal{M}_n)\, p(\mathcal{M}_n)}{p(d_k)}$$

In terms of the mixture model notation here, this becomes

$$q_n(\vec{x}_k) \;=\; \frac{m_n\, p_n(\vec{c}_k \,|\, \vec{a}_n)}{\sum_{n=1}^{N} m_n\, p_n(\vec{c}_k \,|\, \vec{a}_n)}\;. \tag{62}$$

That is, the likelihood of the constraint given the n^{th} model is simply $p_n(\vec{c}_k \,|\, \vec{a}_n)$, and the probability of the n^{th} model is just m_n. The denominator is just the marginalization of the joint distribution $p(\vec{c}_k,\, \vec{a}_n)$ over the space of models. And of course it is easy to show that

$$\sum_{n=1}^{N} q_n(\vec{c}_k) = 1 \tag{63}$$

In the context of the EM algorithm these ownership probabilities can be viewed as soft assignments of data to models. Once these assignments are made we can perform a weighted regression to find the motion parameters of each model.

Mixture Model Estimation. To estimate the model parameters and ownership probabilities our goal is to find extrema of $\log L(\vec{m},\, \vec{a}_1,\, ...,\, \vec{a}_N)$ subject to the constraint that $||\vec{m}|| = 1$. At local extrema, it has been shown that the following must be satisfied:

$$\frac{1}{K}\sum_{k=1}^{K} q_n(\vec{c}_k) \;=\; m_n \tag{64}$$

$$\sum_{k=1}^{K} q_n(\vec{c}_k)\frac{\partial}{\partial \vec{a}_n}\,\log p_n(\vec{c}_k \,|\, \vec{a}_n) \;=\; \vec{0} \tag{65}$$

These two equations follow from setting the derivative of the log likelihood function (with respect to $\vec{m}$ and $\vec{a}_n$) to zero, along with a Lagrange multiplier to ensure that $||\vec{m}|| = 1$. The first equation, Eq. (64), follows directly from Eqs. (62) and (63) and the constraint that $||\vec{m}|| = 1$. The second equation, Eq. (65), specifies conditions for the maxima of the log likelihood with respect to model parameters. A detailed derivation of these conditions is beyond the scope of this introduction.

Because the mixture model likelihood (objective) function here will have multiple local minima, a starting point for the EM iterations is required. That is, to begin the iterative procedure one needs an initial guess of either the ownership probabilities, or of the model parameters (motion and mixture parameters). Often one starts by choosing random values for the initial ownership probabilities and then begin with the estimation of the mixing probabilities and the motion model parameters.

Given ownership probabilities, the estimation of the mixing probabilities is given simply by Eqn. (64). In other words the mixing probability for model $\mathcal{M}_n$ is just the fraction of the total available ownership probability assigned to the n^{th} model. The estimation of the motion model parameters is similarly straightforward. That is, given the ownership probabilities, we can estimate the motion parameters for each model independently. The ML estimation of the motion parameters of the n^{th} model is an weighted area-based regression problem. For the case of a translational motion model, where the motion parameters are just $\vec{a}_n \equiv \vec{\mathbf{u}}_n$, this is just the minimization of the weighted

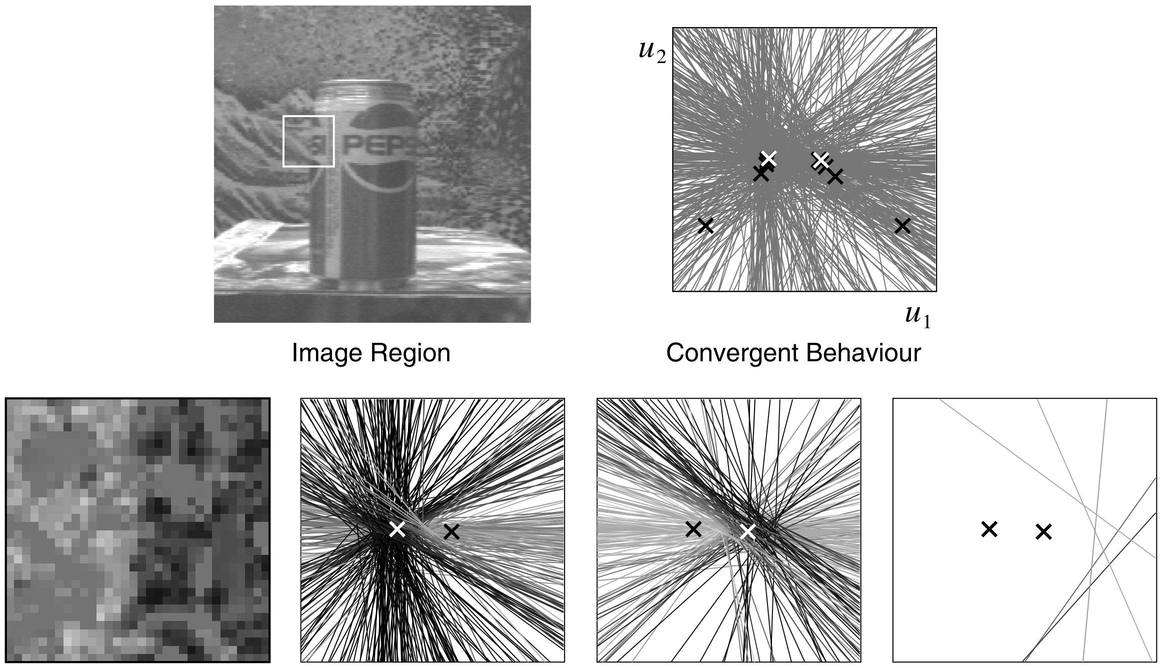


Figure 14: A region containing a depth discontinuity contains two main populations of constraints, consistent with one or other of the two motions. Applying the EM algorithm to this region, The black crosses in the upper right show the sequence of motion estimates at each iteration. The white crosses show the estimates at convergence. With these parameters, the bottom figures show ownership probabilities. The left image shows ownership probabilities of each pixel (based on the motion constraint at that pixel). The next two plots shown the velocity constraints where intensity depicts ownership (black means a high ownership probability). The far right figure shows the constraints owned by the outlier model.

---

least-squares error

$$\begin{aligned} E(\vec{\mathbf{u}}_n) &= \sum_{k=1}^{K} q_n(\vec{c}_k)\, d^2(\vec{\mathbf{u}}_n,\, \vec{c}_k) \\ &= \sum_{k=1}^{K} q_n(\vec{c}_k)\, \left[\vec{\nabla} f(\vec{x}_k, t) \cdot \vec{\mathbf{u}}_n + f_t(\vec{x}_k, t) \right]^2 \ , \end{aligned} \tag{66}$$

where $\vec{\nabla} f \equiv (f_x,\, f_y)^T$. This least-squares estimate is identical to that given above in Eq. (20) in Section 2, with the window weighting function replaced by $q_n(\vec{c}_k)$.

Outliers. One problem with the example here is that the gradient constraint errors are not always expected to be Gaussian (i.e., well-behaved). In particular, gradient measurements near an occlusion boundary may not be consistent with either one of the two motions. Rather, they may be considered outliers, producing errors that lie far from the typical errors distributions we expect. As a result, we might introduce an outlier model, $\mathcal{M}_0$, in addition to the two motion models.

We'll assume that the measurement errors are well described by a Gaussian process when they are small (i.e., consistent with one of the two motions). Otherwise, following Jepson and Black (1993), let's assume that the remaining measurements should be owned by $\mathcal{M}_0$. Moreover, let's model the outlier probability as uniform, where the probability of observing a constraint $\vec{c}_k$, conditioned on being an outlier, is a constant p_0.

The results shown in Figure 14 were computed with two motion models and an outlier model like that described here. For the region shown in Figure 14, the measurement constraints owned by the outlier model are shown in the bottom-right plot.

Remaining Issues. So far we have just touched on some of the main ideas on mixture models, the EM algorithm, and their application to motion estimation. Some of the other interesting questions concern how one might choose appropriate values for the error distributions σ_v, how one selects the number of models for a given image region, how to introduce spatial coherence constraints on the ownership probabilities. Some good references on mixture models include (Ayer and Sawhney, 1995; Jepson and Black, 1993; Weiss, 1996,1997).

References

[1] E. H. Adelson and J. R. Bergen. Spatiotemporal energy models for the perception of motion. *Journal of the Optical Society of America A*, 2:284–299, 1985.

[2] E. H. Adelson and J. R. Bergen. The extraction of spatio-temporal energy in human and machine vision. In *Proceedings of IEEE Workshop on Motion: Representation and Analysis*, pages 151–156, Charleston, S Carolina, 1986.

[3] J K Aggarwal and N Nandhakumar. On the computation of motion from sequences of images — a review. *Proceedings of the IEEE*, 76:917–935, 1988.

[4] P. Anandan. A computational framework and an algorithm for the measurement of visual motion. *International Journal of Computer Vision*, 2:283–310, 1989.

[5] S. Ayer and H. Sawhney. Layered representation of motion video using robust maximum-likelihood estimation of mixture models and MDL encoding. In *IEEE International Conference on Computer Vision*, pages 777–784, Boston, MA, 1995. IEEE.

[6] J Barron. A survey of approaches for determining optic flow, environmental layout and ego-motion. Technical Report RBCV-TR-84-5, Department of Computer Science, University of Toronto, 1984.

[7] J. L. Barron, D. J. Fleet, and S. S. Beauchemin. Performance of optical flow techniques. *International Journal of Computer Vision*, 12(1):43–77, 1994.

[8] J. R. Bergen, P. Anandan, K. Hanna, and R. Hingorani. Hierarchical model-based motion estimation. In *Proceedings of Second European Conf. on Comp. Vis.*, pages 237–252. Springer-Verlag, 1992.

[9] M. J. Black and P. Anandan. The robust estimation of multiple motions: Parametric and piecewise-smooth flow fields. *Computer Vision and Image Understanding*, 63:75–104, 1996.

[10] M. J. Black and A. D. Jepson. Estimating optical flow in segmented images using variable-order parametric models with local deformations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(10):972–986, October 1996.

[11] C. Bregler and J. Malik. Tracking people with twists and exponential maps. In *Proc. Computer Vision and Pattern Recognition, CVPR-98*, pages 8–15, Santa Barbara, June 1998.

[12] A R Bruss and B K P Horn. Passive navigation. *Computer Vision, Graphics, and Image Processing*, 21:3–20, 1983.

[13] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society Series B*, pages 1–38, 1977.

[14] D. J. Fleet. Disparity from local weighted phase-correlation. In *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, pages 48–56, San Antonio, 1994.

[15] D. J. Fleet, M. J. Black, Y. Yacoob, and A. D. Jepson. Design and use of linear models for image motion analysis. *International Journal of Computer Vision*, 36(3):169–191, 2000.

[16] D. J. Fleet and A. D. Jepson. Computation of component image velocity from local phase information. *International Journal of Computer Vision*, 5:77–104, 1990.

[17] D. J. Fleet and A. D. Jepson. Stability of phase information. *IEEE Pattern Analysis and Machine Intelligence*, 15:1253–1268, 1993.

[18] D. J. Fleet, A. D. Jepson, and M. Jenkin. Phase-based disparity measurement. *Computer Vision, Graphics, and Image Processing*, 53:198–210, 1991.

[19] D.J. Fleet. *Measurement of Image Velocity.* Kluwer, Norwell, MA, 1992.

[20] W. Freeman and E. H. Adelson. The design and use of steerable filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13:891–906, 1991.

[21] N. M. Grzywacz and A. L. Yuille. A model for the estimate of local image velocity by cells in the visual cortex. *Proceedings of the Royal Society of London A*, 239:129–161, 1990.

[22] D J Heeger. Model for the extraction of image flow. *Journal of the Optical Society of America A*, 4:1455–1471, 1987.

[23] D J Heeger and A D Jepson. Subspace methods for recovering rigid motion I: Algorithm and implementation. *International Journal of Computer Vision*, 7:95–117, 1992.

[24] B. K. P. Horn and B. G. Schunk. Determining optical flow. *Artificial Intelligence*, 17:185–203, 1981.

[25] A. Jepson and M. J. Black. Mixture models for optical flow computation. In *Proc. Computer Vision and Pattern Recognition, CVPR-93*, pages 760–761, New York, June 1993.

[26] S. X. Ju, M. J. Black, and A. D. Jepson. Skin and bones: Multi-layer, locally affine, optical flow and regularization with transparency. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 307–314, San Francisco, June 1996.

[27] C. Kuglin and D. Hines. The phase correlation image alignment method. In *Proc. IEEE International Conference of the Cybernetic Society*, pages 163–165, 1975.

[28] H C Longuet-Higgins and K Prazdny. The interpretation of a moving retinal image. *Proceedings of the Royal Society of London B*, 208:385–397, 1980.

[29] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence*, pages 674–679, Vancouver, 1981.

[30] G. J. McLachlan and K. E. Basford. *Mixture Models: Inference and Applications to Clustering.* Marcel Dekker Inc., N.Y., 1988.

[31] H. H. Nagel. On the estimation of optical flow: relations between different approaches and some new results. *Artificial Intelligence*, 33:299–324, 1987.

[32] O. Nestares and D. J. Fleet. Likelihood functions for general error-in-variables problems. In *Proceedings of the IEEE International Conference on Image Processing*, page submitted, Barcelona, Spain, 2003.

[33] O. Nestares, D. J. Fleet, and D. J. Heeger. Likelihood functions and confidence bounds for total-least-squares estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 523–530, Hilton Head, South Carloina, 2000.

[34] E. P. Simoncelli. *Distributed representation and analysis of visual motion.* PhD thesis, Electrical Engineering Department, MIT, 1993.

[35] E P Simoncelli, E H Adelson, and D J Heeger. Probability distributions of optical flow. In *Proceedings of Computer Vision and Pattern Recognition*, pages 310–315, Mauii, HI, June 1991.

[36] A. Verri, F. Girosi, and V. Torre. Differential techniques for optical flow. *Journal of the Optical Society of America A*, 7:912–922, 1990.

[37] J. Y. A. Wang and E. H. Adelson. Representing moving images with layers. *IEEE Transactions on Image Processing*, 3(5):625–638, September 1994.

[38] A B Watson and A J Ahumada. A look at motion in the frequency domain. In J K Tsotsos, editor, *Motion: Perception and representation*, pages 1–10. Association for Computing Machinery, New York, 1983.

[39] A. B. Watson and A. J. Ahumada. Model of human visual-motion sensing. *Journal of the Optical Society of America A*, 2:322–342, 1985.

[40] A M Waxman and S Ullman. Surface structure and three-dimensional motion from image flow kinematics. *International Journal of Robotics Research*, 4:72–94, 1985.

[41] J. Weber and J. Malik. Robust computation of optical-flow in a multiscale differential framework. *International Journal of Computer Vision*, 14(1):67–81, January 1995.

[42] Y. Weiss. Smoothness in layers: Motion segmentation using nonparametric mixture estimation. In *Proceedings of IEEE conference on Computer Vision and Pattern Recognition*, pages 520–526, Puerto Rico, June 1997.

[43] Y. Weiss and E. H. Adelson. A unified mixture framework for motion segmentation: Incorporating spatial coherence and estimating the number of models. In *Proc. Computer Vision and Pattern Recognition, CVPR-96*, pages 321–326, San Francisco, June 1996.